Dragon Age: Origins (DAO)

From Woe to Go:
How to make a mod using the DAO toolset

Stephen Karpinskyj (WeVEn) and Anna Tito (MythicalC)

This tutorial set was originally developed by us for the RMIT Games Studio 1 Class. You will notice two very different styles of tutorials in this series. The Art and Design side of the toolset has been covered by Anna and the Programming and Technical 101 side has been covered by Stephen.

We would like to thank the builders of the Dragon Age Wiki for all the resources they have supplied as well as the numerous members of the modding community who have written or recorded fantastic DAO tutorials.

If you have any questions, feedback or notice any errors please let us know by e-mailing Anna at anna.tito@mythicalcreature.net.

Have fun modding ☺

Table of Contents

INTRODUCTION

Important Resources

- Dragon Age Builder Wiki: http://social.bioware.com/wiki/datoolset/index.php/Main_Page
- BioWare Social Network Particularly the DA:Builder Utilities Projects http://social.bioware.com/browse_bw_projects.php?page_num=1&project_search=Search&view=0&project_category_id=6&sort=1
- YouTube video tutorials particularly those by the following users:
    - dragonage22, link to Tutorial 1: http://youtu.be/DKJ7F14n8o8
    - St4rdog, link to Tutorial 1:http://youtu.be/TwQNRBFLhrE

Installing the Toolset

If you want the toolset at home, you will need a buy a copy of the game and register it with a BioWare account, you can then download the toolset (http://social.bioware.com/page/da-toolset).

There are a few tricks to getting the mod installed and fully working.

1. Make sure you install all the recommended files, the toolset is very version specific and you will need to make sure the right version of Microsoft SQL Server Express and all the Core Resources are installed.
2. For light mapping and level exporting to work you will need:
    a. Python 2.5.4 available here: http://www.python.org/ftp/python/2.5.4/python-2.5.4.msi
    b. The win32 extensions for Python 2.5 available here http://sourceforge.net/projects/pywin32/files/pywin32/Build%20214/pywin32-214.win32-py2.5.exe/download
    OR
    The ActiveState Python 2.5.5.7 available here: http://downloads.activestate.com/ActivePython/releases/2.5.5.7/ActivePython-2.5.5.7-win32-x86.msi

I personally Used the win32 extensions and it worked fine, I installed it as all 32bit because it can have issue with the 64. The DA Builder Wiki has an excellent 'How to' and known bugs and solutions section see: http://social.bioware.com/wiki/datoolset/index.php/Installing_the_toolset

File Naming Convention

DA: O has a very strong resource and module naming convention, it is a good idea to follow it when building your mod. You can find the documentation on naming conventions here:
http://social.bioware.com/wiki/datoolset/index.php/Naming_conventions

Resource Naming Convention

Naming conventions information from the DA Builders Wiki (link above):

cir200cr_bloodmage

1. The three letter prefix (green) in this case "cir," this indicates which large scale area the object is from in your case it will be your mod prefix e.g. My mod was called "A Little Red" so we would use the prefix "alr" for our files. However in a larger mod you may want to use a set prefix for a certain area or quest tree. For instance if you have a section of the game that is in mines then using 'min' as a prefix would help you to separate out your assets. There are a number of prefixes already in use which you can see on the Prefixes in use page on the DA Builders Wiki, It is important not to use any of these as it will cause conflicts with the Core resources of the Game or its expansions and modules: http://social.bioware.com/wiki/datoolset/index.php/Prefixes_in_use
2. three-digit number unique to a particular area, generally starting with 100 and incrementing in hundreds for major areas. Accessory areas are given numbers within the block of one hundred (usually multiples of 10) that their 'parent' area belongs to. "global" resources are often given the number 000. E.g. if you had a chair for your castle level and the castle was "alr100ar_Castle" then your chair would be alr110ip_Chair, this tells you that it is a Chair placeable , in the castle area of the mod "A Little Red".
3. two-letter code indicating what general type of resource it is ("ar" for area, "al" for area list, "ip" for placeable, "cr" for creature, "im" for item, "tr" for trigger, "pt" for plot, "st" for stage, "cs" for cutscene, "wp" for waypoint). A major exception here is event scripts, which have exactly the same name as the resource they're the script for. Dialogue files are also generally named exactly the same as their owner resource. Merchants usually have the owner name, with the prefix "store_".
4. An underscore.
5. The remainder of the name is free-form and human-readable, a descriptive term to remind you at a glance what the resource is.

If you are looking at developing your own models, voice over, animation or vegetation for your mod there are also specific model naming conventions for each of these object types which you can find on the Naming Conventions page listed above.

Database Introduction

This week you will be learning how the DA:O resource database works so that you can store and delete all of your resources (e.g., areas, scripts, creature templates, etc).

Manage your resources in the database

Most of the ways that you can manage your module's resources in the Dragon Age database (e.g., check in/out, delete, undo, etc).

Resources that you create in the toolset are automatically added to your computer's copy of the Dragon Age resource database. Resources must be checked out of the database before they can be edited. New resources are, by default, "checked out".

If you are sharing a database, only one person can have a checked out/editable copy of a resource at any one time.

1. **Check out** one of your resources from the database by right-clicking on it in the "Palette Window" and selecting "Check Out" (its icon will now look like).

Once you have finished editing a new version of a checked out resource, it can be "checked in" again (i.e., the database will be updated with your new version and you can only view a read-only copy of the resource).

2. **Check in** one of your resources that has already been checked out by right-clicking on it in the "Palette Window" and selecting "Check In" (its icon will now look like).

The changes made to checked out resource can also be "undone" (i.e., any changes you made to the resource since it was last checked out will be lost and the resource will become checked in again).

Undoing a checkout is permanent! Once you undo your changes, they're really gone!

3. **Undo all changes** to a resource since it has been last checked out by right-clicking on it in the "Palette Window" and selecting "Undo Checkout".

A checked out copy of your resource cannot be deleted, your edited copy must be checked in or be undone before the delete option becomes available.

4. **Delete** one of your resources from the database by making sure it is checked in (i.e., its icon looks like) and right-clicking on it in the "Palette Window" and selecting "Delete".

To look at a history of resources that you have deleted, select "Tools => Deleted Resources" from the top menu bar.

5. **View changes between** your currently edited version and the last checked in version by right-clicking on it in the "Palette Window" and selecting "Diff to Last Checked In".

6. **View the change history** of one of your resources by right-clicking on it in the "Palette Window" and selecting "Resource History".

You can also open and edit a "local copy" of a resource. These are always separated from the database and any changes made to a local copy are not saved if the toolset is closed.

7. **Open a local copy** of one of your resources by making sure it is checked in (i.e., its icon looks like 🛢) and right-clicking on it in the "Palette Window" and selecting "Open Local Copy".

The purpose of using a database to manage resources is twofold: 1) prevent team members from overwriting each other's changes and 2) protect a working copy of your work from any experimention or file corruption.

You've finished the database introduction and learnt how to add, edit, update and delete resources using database terminology such as check in, check out and undo. If any or all of the above wasn't clear, see the builder wiki's brief explanation of checking in/out (link) **or ask questions in class, this is important stuff!**

File system and Resource Introduction & Submitting Modules

This week you will be learning where to place all of your resources, whether they are uncompiled or compiled as well as defining what uncompiled, compiled, designer, art and 2DA resources are.

The point of this tutorial is so you can use the "Builder to Builder" and "Builder to Player" tools to submit your modules as part of your assignments.

Understanding the DA:O filesystem

In order to better understand where you should place special files that you add to your module or to load other modules, you need to get to know the convoluted Dragon Age file system.

Your **"module folder"** is automatically created when you add a module in the toolset and is named after your module's "UID" property. It is located in your current Windows profile's "My Documents" or "Documents" folder:

"<Documents> \ BioWare \ Dragon Age \ Addins \ <Module Name>"

Your **"module's override folder"** is where you should place all compiled assets that you wish to add or use to overwrite existing assets (e.g., exported level layouts, .GDA files, etc). Resources placed here can be read by your module (including when your module is open in the toolset) but not by other modules (including when another module is open in the toolset). It is located in your module folder:

"<Module Folder> \ module \ override"

Your **"module's assets folders"** are where you should place all uncompiled resources (e.g., uncompiled art, sound or 2DA assets (e.g., unexported levels, .WAV files, .XLS files, etc). This is not required but will make it much easier to create a package of all your work when submitting your assignments. This set of folders are located in your module folder:

"<Module Folder> \ assets \ 2da"
"<Module Folder> \ assets \ art"
"<Module Folder> \ assets \ sound"

Your **"Windows profile's override folder"** is where module files (.DAZIP packages) are placed when installed (see section 4). Other compiled resources can be copied here if you want them to be accessible from <u>any module</u>. It is located in your "My Documents" or "Documents":

"<Documents> \ BioWare \ Dragon Age \ packages \ core \ override"

It is not as important to know what other override folders are used for so don't read the following explanations unless you're really keen!

Your **"module's core override folder"** is where you place resources that you wish to be accessible from the current module **and other modules**. This folder is used for resources that have the "Owner Module" property set as "Core" instead of the current module. This is because not all modules are standalone campaigns, sometimes they are modifications for other campaigns, including the official single-player campaign. <u>You can disable/enable the effect of modules in the DA:O main menu by selecting "Downloadable Content => Installed Content"</u>.[1] It is located in your module folder:

"<Module Folder> \ core \ override"

Your **"computer's override folder"** is exactly the same as your Windows profile's override folder except that it **affects all Windows profiles**, not just the current. It is located in the DA:O install directory:

"<DA:O Install Directory> \ packages \ core \ override"

Understanding DA:O resources

Resources that you create inside the toolset (e.g., scripts, areas, levels, 2DA's, etc) or outside the toolset (e.g., textures, models, GDA's) are either compiled or uncompiled.

You should place all resources in their <u>correct</u> place as explained during this section!

The term **uncompiled resources** applies to "raw" or "source" resources that have not yet gone through a conversion process and can generally, but not always, be opened with programs other than the toolset (e.g., .WAV, .XLS files).

The term **compiled resources** applies to "exported" or "built" resources that either the toolset, external applications (e.g., FMOD) or a plugin for other applications (e.g., 3D Studio Max) converts to another filetype that is optimised to be quickly loaded into your computer's memory and read by DA:O.

However, not <u>all</u> resources need to be compiled and are supported by the toolset in their native format (e.g., .DDS is a supported texture format that can also be edited in Adobe Photoshop, GIMP, etc). See the builder wiki for more info on .DDS files (link).

In order to create custom content for your modules you are nearly always going to have to follow a resource compilation processes of some kind. Some common processes are as follows:

1. Using the toolset to export level files (.LVL) as area layouts.
2. Using FMOD to convert a project of uncompiled .WAV audio files into compiled .FSB/.FEV files. See the builder wiki for more info (link).

[1]As far as I know, this is the difference between the Window's profile override folder and a module's core override folder. Although they both affect more than a particular module, you can't disable/enable modifications if they are added to the Window's profile override folder and therefore not associated with a particular module.

3. Using ExcelProcessor to convert uncompiled .XLS 2DA files into compiled .GDA files. See the builder wiki for more info (link).

4. Using a plugin for 3D Studio Max to export meshes as .MMH/.MSH files. See the builder wiki for more info (link).

Resources generally fall into one of the following 3 categories:

1. "Designer resources" are always created and edited in the toolset (e.g., areas, item templates, creature templates, trigger templates, etc). When you create new or override existing designers resources in the toolset, they primarily exist in your computer's DA:O database and don't have an uncompiled form. For more info, see the builder wiki (link).

2. "Art resources" form the *physical* aspect (i.e., appearance/sound) of designer resources like models, animations, etc (e.g., A designer resource such as an item is a model plus a set of editable properties. You must place all your **compiled art resources** in your module's override folder (e.g., MM). For more info, see the builder wiki (link).

It is strongly recommended that all uncompiled art resources are placed in your module's art asset directory (see section 1). This is not required but will make your work easier to package for submission.

3. "2DA resources" form the data that is often either *attached* to designer resources (e.g., a trigger template has an attached variable 2DA) or used to *reference* art resources (e.g., a list of visual effects). You must place all your **compiled 2DA resources** in your module's override folder (e.g., ".GDA" files). For more info, see the builder wiki (link).

It is strongly recommended that all uncompiled 2DA resources are placed in your module's 2DA asset directory (see section 1). This is not required but will make your work easier to package for submission.

Lastly, your are likely to end up with a lot of resources in your module's override folder so you have 2 ways of keeping them neat and tidy:

1. Subfolders can be placed into your override folder (e.g., a subfolder to place 2DA files, another for voice-over files, etc).

2. .ERF files are resource packages that the toolset can create/extract and DA:O will read. You can package up your resources (e.g., a package for your creatures, etc). See the builder wiki for more uses for .ERF's (link).

Using the "Builder to Builder" tools

The steps you need to take to package some/all of your module's uncompiled resources so that they can be transferred to another computer's resource database.

In order to transfer your uncompiled resources from one computer to another (i.e., transfer to home/uni or pass your work to your teammates) you will need to use the toolset's "Builder to Builder Create" tool.

- In the "Palette Window", highlight the resources you wish to transfer to another computer.
- Right-click the selected resources and select "Tools => Builder => Builder to Builder Create".

- Press "Ok" in the "Builder to Builder Create" tool to open the "Save As" dialog box.
- Find a place on your computer to store your work, name the package and press "Save".

In order to add uncompiled resources from another computer (i.e., transfer from home/uni or load your teammate's work) you will need to use the toolset's "Builder to Builder Load" tool.

- Select "Tools => Builder => Builder to Builder Load" in the top menu bar to open the tool.
- Find the package (.DADBDATA file) you want to load (e.g., on a USB drive) and press "Open".
- Check/tick the boxes of the resources you wish to import into your database and press "Ok". This will either add a new resource to the database or overwrite the checked in version of an existing resource.

The "Builder to Builder Load" tool won't automatically check in the resource that you imported if it is a resource that already exists in the database. It is up to you whether you check in a file to your database after importing an existing resource.

You can press "Check In Resource" from within the "Builder to Builder Load" tool if you want to update one or more of your resources in the database before you overwrite it.

Make sure "Create New" from the "StringIDs" section of the "Builder to Builder Load" tool is selected. This will avoid a toolset bug that often doesn't load resources correctly but it's a big pain to remember when "Use Theirs" is always selected by default!

For more information on the "Builder to Builder" tools, see the builder wiki (link**).**

Using the "Builder to Player" tool

The steps you need to take to package a compiled version of your module so that it can be played on another computer.

In order to transfer a playable version of your module's resources (i.e., compiled resources) you should use the toolset's "Builder to Player Package" tool.

1. Open the Dragon Age Toolset and make sure your module is open.
2. In the top menu bar, select "Tools => Builder => Builder to Player Package".
3. When prompted to open a "user manifest", either select a previous one or press "Cancel".
4. In the "Builder to Player Package" dialog, select your module files. If you have kept all your module resources in your module's directory (see section 1) then this involves unselecting all files and only checking your module folder.
5. With all your module files checked, press "Ok" to open the "Save As" dialog box.
6. Select a directory and name for your module package and press "Save".
7. You have the option to save a user manifest (i.e., xml file containing filenames and directories) of the files you just added to your module package.

You can open this user manifest in future to automatically check the same list of module resources.

For more info on the "Builder to Player" tool, see the builder wiki (link).

8. To install a module package (i.e., a .DAZIP file), open the "DAUpdater" application found in "<DA:O Install Directory> \ bin_ship".

9. Press the "Select DAZips" button, browse your computer for the module package to install and press "Open" to go back to the DAUpdater application.

10. Select that same module package from the module list and press "Install Selected".

DAUpdater will, amongst other things, copy the new module to your Windows profile's override folder (for more info, see section 1).

Submitting your modules

The steps you will need to take to submit both an uncompiled and compiled version of all your module's resources for future assignments.

Unfortunately there is no one-step method for packaging all resources required for submission. The method below is still quite tedious and easy to stuff up!

1. Create a package of your module's **uncompiled designer resources** by selecting all your designer resources in the toolset's palette window, right-clicking them and selecting "Builder to Builder Create" (see section 3 for how to use the "Builder to Builder Create" tool). The tool will produce a .DADBDATA file for you to include in your submission.

2. Create a package of your module's **uncompiled art and 2DA resources** by adding them all to a .zip file. If you followed previous advice, all these resources should be in your module's assets folders (see section 1 and 2). Add all the assets folders (2DA, art and sound) to a .ZIP file and name it as "assets.zip" or similar.

To make doubly sure your submission contains everything, we recommend zipping up your entire module folder, and not just the assets folder.

3. Create a package of all your module's **compiled resources (designer, art and 2DA)** by exporting the latest version of your resources in the toolset[2] and using the Builder to Player tool to create a package that includes all your playable module files (see section 4 for how to use the "Builder to Player" tool). The tool will produce a .DAZIP file for you to include in your submission.

4. When submitting, make sure you have copied all of the following files to your disc:

- <Module Name>.dadbdata
- Asset folders or entire module folder (.ZIP)
- <Module Name>.dazip
- Design document (.PDF)

[2]See scripting tutorial 2, section 4 if you're not sure what "exporting resources" means.

SCRIPTING

Scripting Tutorial 1 – Scripting Intro

This week you will be learning how to set up a new module with a test area in the Dragon Age Toolset and then attach a basic script to your area.

Create a new test module

The basics steps you will need to take to create a new test module once you have installed the Dragon Age Toolset.

1. Open the Dragon Age Toolset.
2. In the top menu bar, select "File => Manage Modules".
3. In the "Manage Modules" dialog box, select "New..." to open the "Object Inspector" dialog box (i.e., the properties of your new mod).
4. Enter the title of your new module into both the "Name" and "UID" fields (e.g., "Tutorials") and click "OK".
5. Back in the "Manage Modules" dialog box, choose your new module from the list and click "Open".

For an advanced look at creating modules, check out the official wiki's page (link).

Deleting a module is not easy and can be a real pain (link) so keep that in mind before creating lots of test modules.

Create a new test area

The steps you will need to take to quickly create an area for testing out your scripts once you have created and opened your new test module.

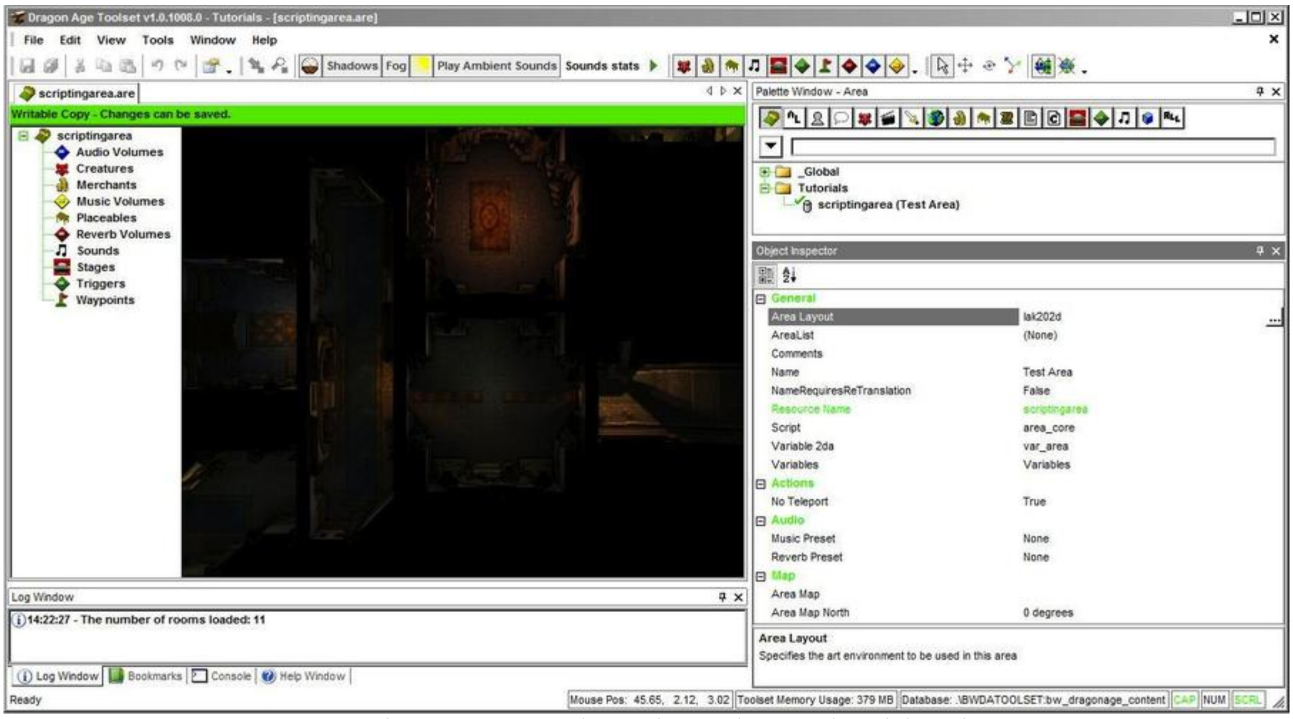
1. Open the Dragon Age Toolset and make sure your your new module is open.

To see which module the toolset has open, look at the end of title bar (very top-left of the tool-set screen). It will read "Dragon Age Toolset v1.0... - <Module_Name>".

2. Remember if your module is not open, select "File => Manage Modules" in the top menu bar, choose your module from the list in the "Manage Modules" dialog box and click "Open".
3. In the top menu, select "File => New => Area" to open the "Create New Resource" dialog box.
4. Enter the name of your test area into the "Resource Name" field (e.g., "scriptingarea").
5. Create a folder to store all of your module's resources by entering a new folder name into the "Folder" field (e.g., "\Tutorials").

Wherever possible, it is a good idea to keep all of the resources for your module in the one re-source folder that has a similar name to that of your module.

6. Press "OK" to close the dialog box and create your test area.
7. With your new area open, look at its editable properties in the "Object Inspector" (bottom-right of the toolset), edit the "Name" field and then click on the "Area Layout" field.
8. Select an area layout by selecting the "..." button and choosing from the list (e.g., "lak202d").


Dragon Age Toolset v1.0.1008.0 - Tutorials - [scriptingarea.are]
File Edit View Tools Window Help
Shadows Fog Play Ambient Sounds Sounds stats
scriptingarea.are
Writable Copy - Changes can be saved.
scriptingarea
Audio Volumes
Creatures
Merchants
Music Volumes
Placeables
Reverb Volumes
Sounds
Stages
Triggers
Waypoints
Palette Window - Area
_Global
Tutorials
scriptingarea (Test Area)
Object Inspector
Log Window
Log Window Bookmarks Console Help Window

14:22:27 - The number of rooms loaded: 11

General	
Area Layout	lak202d
AreaList	(None)
Comments	
Name	Test Area
NameRequiresReTranslation	False
Resource Name	scriptingarea
Script	area_core
Variable 2da	var_area
Variables	Variables
Actions	
No Teleport	True
Audio	
Music Preset	None
Reverb Preset	None
Map	
Area Map	
Area Map North	0 degrees

Area Layout
Specifies the art environment to be used in this area

Ready | Mouse Pos: 45.65, 2.12, 3.02 | Toolset Memory Usage: 379 MB | Database: .\BWDATOOLSET:bw_dragonage_content | CAP | NUM | SCRL

Figure 1: The Dragon Age toolset with a newly created module and area.

Create a starting point for your module

The steps you will need to take to tell your module where the player should begin when the module is played.

1. Ensure your toolset looks something like Figure 1.
2. Right-click in the toolset's environment window (the 3D view of your area) and select "Insert Waypoint".
3. Left-click on an empty space of the area to place your waypoint but keep in mind that you can move your waypoint around to wherever you like after it has been created and placed.

A waypoint is simply a "special point" of your map where things will happen. It might be the start location of an area, a point where you spawn an object with a script, etc.

4. Left-click on your waypoint and look at its editable properties in the "Object Inspector".
5. Enter the name of your waypoint into the "Tag" field of its properties (e.g., "player_start").
6. Open your module's properties ("File => Manage Modules => Properties...").
7. Edit the "Starting Area" field by clicking on it, selecting the "..." button and browsing for your area in your module's resource directory (e.g., in "Tutorials" folder, select "scriptingarea").
8. Edit the "Starting Waypoint" field by clicking on it and selecting your waypoint from the list (e.g., "player_start").
9. Press "OK" to close the properties and then "Close" to return to editing your area.
10. Press "CTRL+S" to save your area.

Playtest your module

The steps you will need to take to export your module's resources (e.g., scripts, areas, etc) in order to play-test the latest version of your module.

1. Select the "ALL" icon on the right of the "Palette Window" (top-left of the toolset).
2. Right-click on your module resource directory (e.g., "Tutorials") and select "Export => Export without dependent resources".
3. Check the "Log Window" (bottom-left of the toolset) for whether your resources were exported successfully.

Selecting "Export *with* dependent resources" is unnecessary and will export the core game resources it has access to as well and this can take a very long time!

4. Run a game of DA:O and select "Other Campaigns" from the game's main menu.
5. You should now see your module in the list (e.g., "Tutorials").

If your computer can handle it, you can leave DA:O running its main menu in the background to save loading the game everytime you want to playtest your module.

Create a new test script

The steps you will need to take to attach an empty script to your newly created test area.

1. With your test area open, select its properties and edit the "Script" field by clicking the "..." button.
2. In the "Resource Open/Save" dialog box, click the "New" button to open the "Create New Resource" dialog box.
3. Enter the name for your script into the "Resource Name" field (e.g., "area_test").
4. Enter the name of your module's resource folder into the "Folder" field (e.g., "\Tutorials").

Another way to add resources automatically into your module's resource folder is to right-click on the folder in the "Palette Window" and select "New => Script/Area/etc".

5. Press "OK" to create your new (and empty) script.

Because your script is attached to an area, your script will execute (i.e., your script will run) every time a game event related to the area is triggered.

6. In the "Palette Window", right-click on your new script and select "Open Resource".
7. Copy and paste the following code into the script's text editor.

```
void main ()
{
    // This is a comment!!!
}
```

The above script does absolutely nothing but keep in mind that commenting any code that is difficult to understand becomes very useful when you share it with teammates.

Compile your new test script

The steps you will need to take to compile your script and make sure it doesn't have any errors.

Compilation is the process of converting human-readable instructions (the code which you write and edit) into computer-readable instructions (a binary file). The compiler will check your code for errors and will report whether the compilation succeeded.

1. The toolset will automatically compile the open script when you save it (i.e., press "CTRL+S").

You can also compile the open script manually, without saving it, by pressing "F7".

2. Check the results of your compilation by reading the topmost entries in the "Log Window" because if/when you get errors, this is where they will be reported to you.

To compile all of your scripts (not just the open script), export all of your module's resources as described above in section 4, steps 1-3.

Make your new test script do something

The steps you will need to take to make your script add an item the player character's inventory every time the script is executed and then playtest the results.

1. Replace line 3 in your script with the following code.

```
3  object oPC = GetMainControlled();
4
5  CreateItemOnObject(R"urn230im_armor_reward.uti", oPC);
```

2. Save your script, re-export your module's resources and then playtest your module again as described above in section 4.

3. While playtesting your module, open your character's inventory by pressing "I" and see the armor that your script created inside the player's inventory.

You've finished this week's tutorial and created a new module, test area and basic script! But you'll notice that your script created 4 items... Next week we'll look at events and why this is the case. You'll also learn how to share your module's resources with teammates and transfer them between computers.

Helpful links

1. A list of DA:O filetypes and what they mean (link).

2. A description of scripting changes that have occurred in the last couple of patches (link).

Scripting Tutorial 2 – Scripting Events

This week you will be learning a tonne of stuff but primarily you will be copying and dissecting 2 scripts that can handle game events.

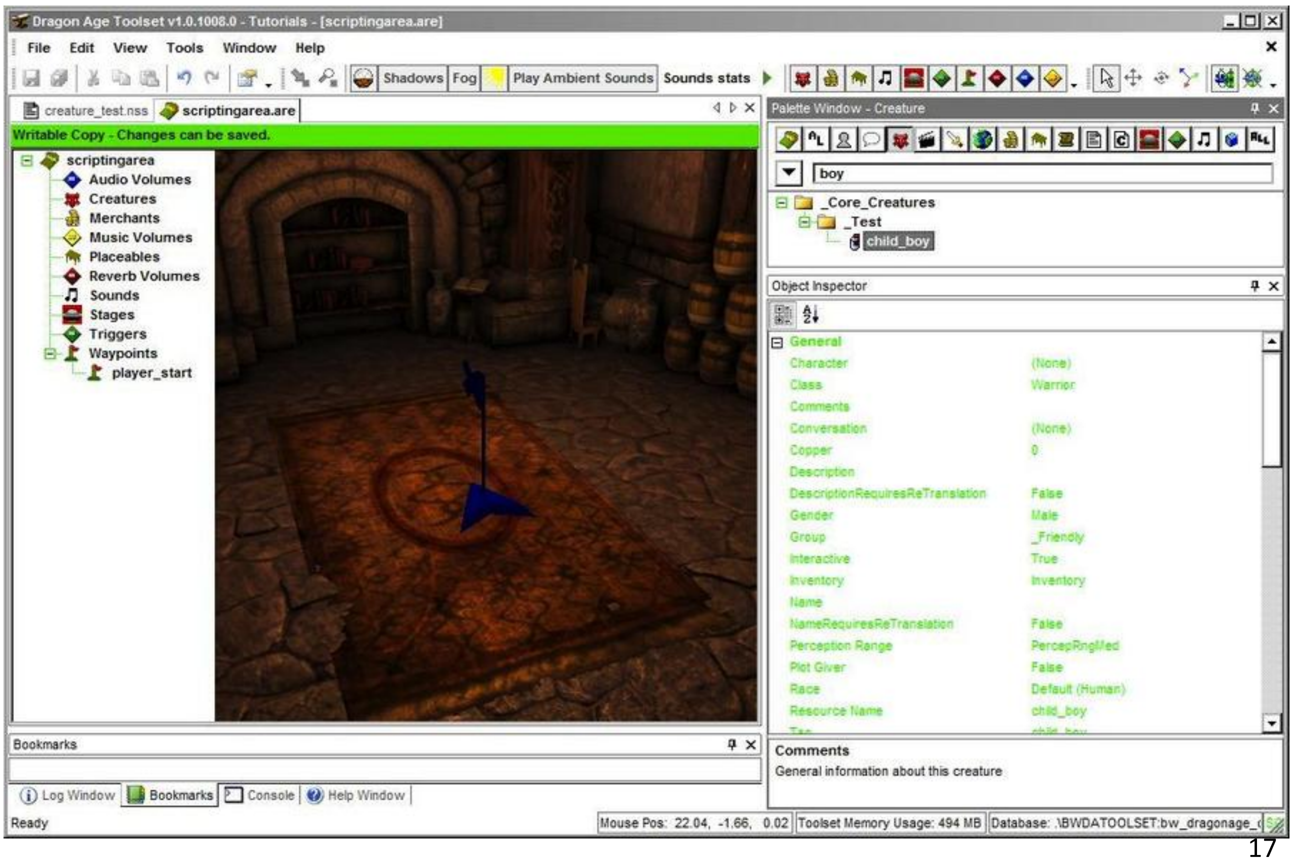
DA:O scripting is all about events. Most of your scripts that you attach to objects will require you to handle different events that trigger the script. These events will be triggered automatically by the game engine, or by actions the player or another object might take (e.g., the player or an NPC might move to a new area, in both cases triggering an event). For a list of most event types, see the builder wiki (link).

Duplicate an existing creature template

The steps you will need to take to duplicate a creature template once you have opened the Dragon Age toolset and loaded the test module you created in tutorial 1.

You cannot edit existing resources so use the following process to duplicate and then customise existing resources of any type (e.g., scripts, item templates, plots, etc).

1. Find and open the existing "child_boy" creature template in the "Palette Window" by selecting the icon and entering "boy" in the filter bar below the icon list (see Figure 1).
2. Right-click the "child_boy" creature template and select "Duplicate" to open the "Duplicate Resource" dialog box.
3. Enter the name of your creature template into the "Resource Name" field (e.g., "boy_test").
4. Select your module's resource folder in the "Folder" field and select your module in the "Module" field.
5. Press "OK" to close the "Duplicate Resource" dialog box and automatically open your new creature template.



General	
Character	(None)
Class	Warrior
Comments	
Conversation	(None)
Copper	0
Description	
DescriptionRequiresReTranslation	False
Gender	Male
Group	_Friendly
Interactive	True
Inventory	Inventory
Name	
NameRequiresReTranslation	False
Perception Range	PercepRngMed
Plot Giver	False
Race	Default (Human)
Resource Name	child_boy

Comments

General information about this creature

Mouse Pos: 22.04, -1.66, 0.02 | Toolset Memory Usage: 494 MB | Database: .\BWDATOOLSET:bw_dragonage_

Instance and customise your new creature template

The steps you will need to take to place an instance (i.e., sort of like a copy) of your creature into your area and then modify the template (i.e., type).

What you see in the "Palette Window" are types of objects (i.e., templates) and what you place into your area are instances of those types. When you modify the type of an object, it will modify all of the instances of that type. This is the same as object-oriented programming where a class is a type of object and what you add to the code are instances of those classes, known as objects that exist at run-time.

1. Make sure you test area is open and your creature template is selected in the "Palette Window".
2. Left-click on an empty space in the area to create an instance of your creature.

Make sure the position you choose for your creature is close to the player's starting waypoint. The reason for this will become obvious later!

3. For fun, open your creature template and edit the "Appearance" field (e.g., select "Pig").
4. Go back to your area and refresh it (i.e., press "F5" or select "View => Refresh" in the top menu bar).

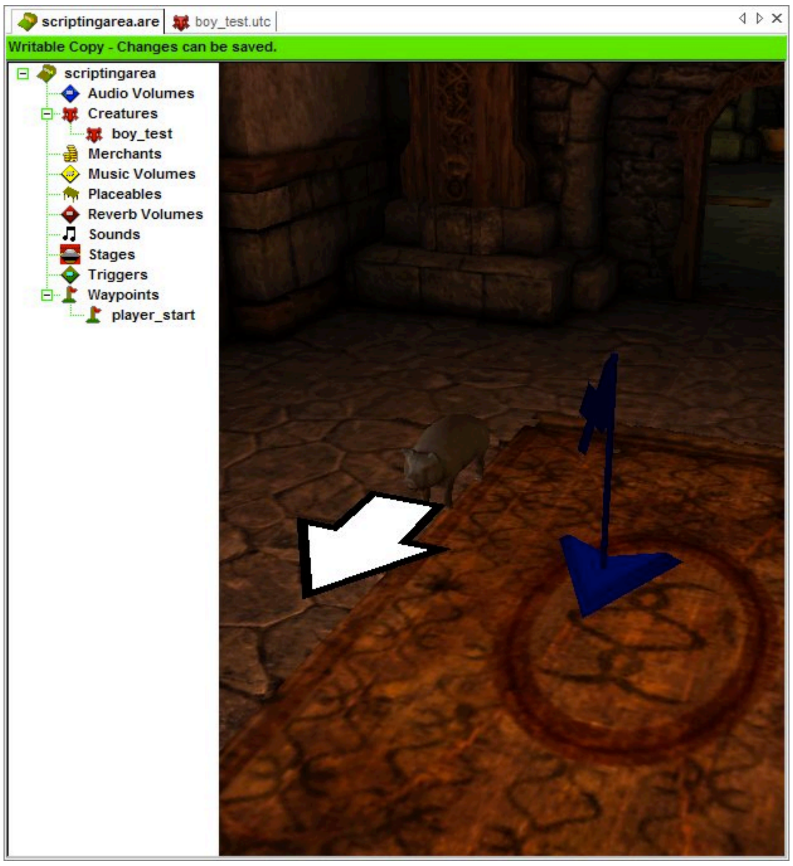


Figure 2: What your area should now look like.

Create a new event-handling script

The steps you will need to take to attach a script to your pig-boy so that it can handle the object's events.

1. Edit the "Script" field of your creature template by creating a new script in it (e.g., name your script "creature_test" (if you can't remember how to do this, review the process detailed in tutorial 1, section 5, steps 1-6).

2. Copy and paste the following code into your new script, compile it and read Explanation 1.

```
 1 void main()
 2 {
 3     event ev = GetCurrentEvent();
 4     int nEventType = GetEventType(ev);
 5
 6     switch (nEventType)
 7     {
 8         case EVENT_TYPE_SPAWN:
 9             // give player item when this creature is created
10             object oPC = GetMainControlled();
11             CreateItemOnObject(R"bec110im_wedding_f.uti", oPC);
12     }
13 }
```

Snippet 1: A script to handle a "spawn" event by creating an item in the player's inventory.

```
 1 The definition of the body of this script.
 2 A symbol defining the start point of this script's body.
 3 Creating and assigning a new variable of type "event". The value as-
   signed here is the event that executes this script.
 4 Creating and assigning a new integer variable. The value assigned here
   is the type of the event variable from line 3.
 6 The definition of a switch statement using the variable from line 4 to
   "control" it. See Wikipedia for more info (link).
 7 A symbol defining the start point of the switch statement.
 8 The first of potentially many case statements inside this switch state-
   ment. Code in the following block (lines 9-11) will only execute/run if
   "nEventType" is equal to "EVENT_TYPE_SPAWN".
 9 Single-line comment to aid those trying to understand the code.
10 Creating and assigning a new variable of type "object". The value as-
   signed here is the object representing the character the player is cur-
   rently controlling (very important line of code!!).
11 Execute an existing function that creates an item (first parameter) in
   the inventory of an object (second parameter).
12 A symbol defining the end point of the switch statement.
13 A symbol defining the end point of this script's body.
```

Explanation 1: A line-by-line explanation of the code in Snippet 1.

If you want more detailed information about Dragon Age scripting and events, see the scripting tutorial on the builder wiki (link).

3. Export your module's resources and playtest your module to see the effects of your code (if you can't remember how to do this, review the process detailed in tutorial 1, section 4, steps 1-5.

Functions were used a lot in Snippet 1 above. Functions are primarily used to either return/"get" variables (e.g., lines 3, 4, 10) or to make something happen (e.g., line 11). A function takes a list of parameters/values so that it knows more specifically what to get or do (e.g., the `GetEventType()` function in line 4 won't know which event type to get unless you supply it with an event instance as the first and only parameter). See Wikipedia for more info on functions (link).

If you're unsure how to use a function, view its definition and extra comments from its author in the "Help Window" by right-clicking on it and selecting "Go To Definition".

Make your new event-handling script do something more exciting

The steps you will need to take to make your event-handling script repeatedly check the distance between the player's character (PC) and the creature executing the script. The script will kill the creature if the player strays too far from the creature.

1. Delete the code you wrote previously in section 3.
2. Copy and paste the following code in its place.

```
#include "utility_h"

void main()
{
    event ev = GetCurrentEvent();
    int nEventType = GetEventType(ev);
    object oPC = GetMainControlled();
    float fDist = GetDistanceBetween(OBJECT_SELF, oPC);

    switch (nEventType)
    {
        case EVENT_TYPE_SPAWN:
            // start the creature's second "timer"
            InitHeartbeat(OBJECT_SELF, 2.0);
            break;

        case EVENT_TYPE_HEARTBEAT2:
            // display how far away player is from creature
            DisplayFloatyMessage(OBJECT_SELF, "Dist=" +
FloatToString(fDist));

            // test if how far away player is from creature
            if (fDist > 8.0)
                KillCreature(OBJECT_SELF);
            break;

        case EVENT_TYPE_DYING:
            DisplayFloatyMessage(OBJECT_SELF, "Oh boy...");
            break;
    }
}
```

Snippet 2: A script to kill the creature it is attached to when the player moves too far away from it.

3. Compile and playtest the new script to see what it does.
4. Try and figure out ***how*** Snippet 2 does what it does using what you just learnt. Any new concepts/keywords you see in Snippet 2 will be explained in future tutorials.

You've finished this week's tutorial and learnt how to create and customise your own creature template, the basics of event handling and some more programming concepts like functions and switch-case statements. Next tutorial we'll look at a "nicer" way of handling events, some more programming concepts like constants and including files. You'll also learn just how the toolset's tag system can make you do some pretty great things.

Scripting Tutorial 3 – Object Variables and 2DA's

This week you will be learning how to create and modify 2DA files as well as how to add and use your own variables that you attach to your own object (i.e., a new trigger template). You will also be writing a script that handles an event properly and dynamically spawns a statute object into your area under certain conditions.

Set up your area and resources

The steps you will need to take to make an area (e.g., your scripting area from previous tutorials) ready for your new script.

1. With an area open in the toolset, create 3 waypoints in an empty patch of floor (see tutorial 1, section 3, steps 1-4 for how to create a waypoint).

If an area is difficult to view in the toolset (e.g., too dark) then turn off the lighting effects by un-selecting "View => Environment => Enable Lighting" in the top menu bar.

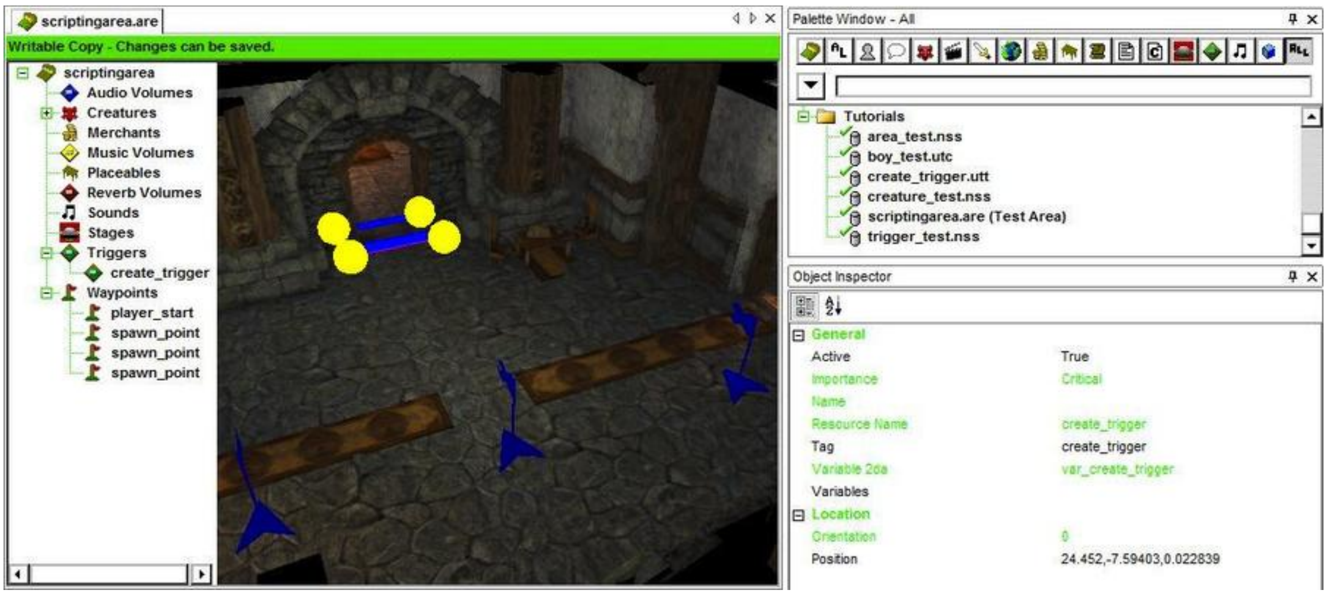
2. Select all 3 waypoints (use shift+click) and edit the "Tag" field (e.g., "spawn_point").

Editing a field with multiple objects selected, will assign your new value to the field of every selected object.

3. Create a new (and empty) script named "trigger_test" or similar.
4. Create a new trigger template named "create_trigger" or similar (if you can't remember how to create a new template, review the process detailed in tutorial 2, section 1, steps 1-5).
5. Change your new trigger template's event script to your new script (e.g., "trigger_test").

You can only modify a template's "Script" property, not an instance's. This is the case for most properties because most of an instance's properties are non-editable. To modify a template's property, open it by double-clicking on the template in the "Palette Window", edit its properties and save the template. These changes are automatically applied to any new or existing instances of that template.

6. Create an instance of your new trigger template near your group of 3 waypoints and save your area. See the builder wiki for more info on triggers and how to create them (link).



General
Active True
Importance Critical
Name
Resource Name create_trigger
Tag create_trigger
Variable 2da var_create_trigger
Variables
Location
Orientation 0
Position 24.452,-7.59403,0.022839

Figure 1: What your area should now look like.

Write your trigger script

The steps you will need to take to write a script that triggers a visual effect at a waypoint's location.

1. Open your empty trigger script (e.g., "trigger_test") and click the "Templates" icon ( ) in the scripting assist window on the right of the code editor.

The script assist also has a list of accessible functions, variables and constants to use.

2. In the script template list, double-click "Custom Trigger Events.txt" to insert a base for your new trigger script.

Notice how the template script is set up to handle an event properly by executing the core trigger script if you don't reassign the "nEventHandled" variable. See the builder wiki for more information about overriding an event script (link).

3. Copy and paste the following code in place of your new script's "EVENT_TYPE_ENTER" case, compile it and then playtest your module.

```
42 case EVENT_TYPE_ENTER:
43 {
44    object oCreature = GetEventCreator(ev);
45
46    // new variables
47    object oWp = GetObjectByTag("spawn_point"); // one of your new waypoints
48    location lWp = GetLocation(oWp);           // location of that waypoint
49    effect eFog = EffectVisualEffect(4031);
50
51    // create the effect
52    Engine_ApplyEffectAtLocation(EFFECT_DURATION_TYPE_TEMPORARY, eFog, lWp);
53
54    // tell the core script not to execute after this
55    nEventHandled = TRUE;
56
57    break;
58 }
```

Snippet 1: A code block to create a visual effect at a waypoint's location.

```
42 The beginning of the block that executes if the switch variable (i.e.,
   in this case "nEventType" from line 27) equals this case (i.e,
   "EVENT_TYPE_ENTER").
43 A symbol defining the start point of this particular case.
47 Creating and assigning a new variable of type "object". The value as-
   signed here is the first object it finds that has the tag
   "spawn_point".
48 Creating and assigning a new variable of type "location". The value as-
   signed here is the location of the waypoint from line 47.
49 Creating and assigning a new variable of type "effect". The mystery
   value assigned here is the ID of a fog-like visual effect.
52 Use the variables created earlier to spawn a visual effect.
55 Set the flag responsible for calling the trigger's core script in line
   71.
57 A statement that causes execution to leave the switch statement (used
   in multi-case switch statements).
58 A symbol defining the end point of this particular case.
```

Explanation 1: A line-by-line explanation of the code in Snippet 1.

Create a new variable 2DA file

The steps you will need to take to add a variable to your own trigger variable 2DA file.

2DA files are nothing more than a simple spreadsheet (.XLS file) with cells of data (e.g., special numbers and strings) organised into titled columns and grouped together in rows. See the builder wiki for more info on 2DA files and how they are used (link).

In this section, we use 2DA files that define a list of object variables that are attached to an object template (e.g., a trigger template).

Object variables (also known as "local variables") are variables that are attached to each instance of a particular object template. Most types of objects have a set of pre-defined variables defined in a 2DA file (e.g., triggers, areas, NPC's, creatures, etc).

Every instance of an object template is attached to an <u>instance</u> of the set of object variables (e.g., all trigger instances of the same type/template do <u>not</u> share variables, each trigger instance has a <u>copy</u> of the variables defined in its template's 2DA).

1. Find where all the object variable 2DA file lives by finding the following folder on your hard-drive: **"<DA:O Install Directory> \ tools \ Source \ 2DA \ toolset"**.

If you purcahased DA:O on Steam, your DA:O install directory will be: "<Steam Install Directory> \ steamapps \ common \ dragon age origins".

2. Copy "var_trigger.xls" to an empty folder (e.g., on the desktop) and open it.
3. Copy row 17 into row 18 and replace "TRIGGER_ROOM_TEXT" with "TIMES_TRIGGERED".
4. Change the name of the worksheet (e.g., "var_create_trigger").

Spreadsheet files are made up of multiple worksheets (also known as "workbooks"). However, in this case the 2DA file only contains one worksheet.

5. Close the spreadsheet file.

Compile a 2DA file

The steps you will need to take to process your new variable 2DA file into a .GDA file using the "ExcelProcessor" tool.

In order for the game to read the data in 2DA files, they need to be processed into a .GDA file. The .GDA file that results from a 2DA is named after the <u>worksheet</u> in the 2DA file (e.g., "var_create_trigger") and <u>not</u> the filename (e.g., "var_trigger"). See the builder wiki for more detailed info on compiling 2DA files (link).

1. Find the "ExcelProcessor.exe" file found in the following folder on your hard-drive: "**<DA:O Install Directory> \ tools \ ResourceBuild \ Processors**".
2. Copy the ExcelProcessor program into the same folder as your new 2DA file.
3. In Windows Explorer, drag your 2DA file onto the ExcelProcessor program to automatically create your .GDA file somewhere inside your current system profile (e.g., on Windows XP, it should create your .GDA file inside: "**C:\Documents and Settings\<Profile Name>**").
4. Copy your new .GDA file to your module's folder on your hard-drive: "**<My Documents> \ BioWare \ Dragon Age \ Addins \ <Module Name> \ module \ override**".
5. Open and close the toolset so you can access your new .GDA file from within the toolset.

Use a .GDA file

The steps you will need to take to make use of your new variable inside your .GDA file.

1. Open your trigger template and edit the "Variable 2DA" property by adding the name of your new .GDA file (e.g., "var_create_trigger").

2. Open your trigger script (e.g., "trigger_test") and replace the code you added in section 2 with the following code:

```
case EVENT_TYPE_ENTER:
{
   object oCreature = GetEventCreator(ev);

   // get and print the value of our new 2DA variable
   int timesTriggered = GetLocalInt(OBJECT_SELF, "TIMES_TRIGGERED");
   DisplayFloatyMessage(oPC, "TIMES_TRIGGERED=" +IntToString(timesTriggered));

   // exit switch-case statement if already been triggered at least 3 times
   if (timesTriggered >= 3)
      break;

   // new variables
   object oWp = GetObjectByTag("spawn_point", timesTriggered);
   location lWp = GetLocation(oWp);
   effect eFog = EffectVisualEffect(4031);

   // create the effect
   Engine_ApplyEffectAtLocation(EFFECT_DURATION_TYPE_TEMPORARY, eFog, lWp);

   // just for fun, spawn a statue
   CreateObject(OBJECT_TYPE_PLACEABLE, R"genip_totem_ds.utp", lWp);

   // increment our new 2DA variable
   SetLocalInt(OBJECT_SELF, "TIMES_TRIGGERED", timesTriggered + 1);

   // tell the core script not to execute after this
   nEventHandled = TRUE;

   break;
}
```

Snippet 2: A code block that gets/sets an object variable. All new code is highlighted.

3. Add the following line after the "HandleEvent" function call:

```
DisplayFloatyMessage(oPC, "handled by core script");
```

4. Compile and playtest the new script to see what it does.

5. To further demonstrate the usefulness of object variables, create a second instance of your trigger template (e.g, "create_trigger") somewhere else in your area.

6. Select your new trigger instance and click on the "..." next to its "Variables" property to open the "Variables" dialog.

7. Change the "Value" of your new "TIMES_TRIGGERED" variable to "2". This will change the default value that this variable is initialised to, each instance can have different initial values!

8. Playtest your module again to see what difference this change makes.

Questions?

1. So where did that magical visual effect ID ("4031") from the earlier sections come from?

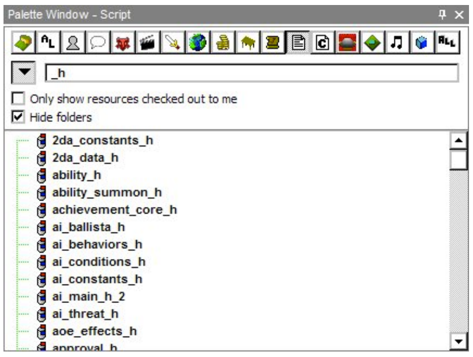
Normally you would be able to use a constant to reference these type of numbers but as far as I know, you can't see the list of visual effects this way. Perhaps because there's so many of them? To see a list of which ID equals which visual effect, go to the following directory on your hard-drive: "<DA:O Install Directory> \ tools \ Source \ 2DA" and open "VFX_base.xls". There's also heaps of other data in this folder that you might find useful.

2. If we use the toolset's "Builder to Bulder" tool to transfer our module between computers will our .GDA file or anything else we place in our module's override folder be automatically copied as well?

No, you need to manually copy all files you place in the override folder <u>in addition to</u> creating a builder package of your module's resources. If you find a better soluion, please share it!

3. How can I find out which header file I have to "#include" (same as "import" in Java) in order to get access to the functions that I need?

As far as I know, there's no quick way to know which header files contain which functions. You can however, view all of the available header files by clicking the script icon (📄) in the palette window, unticking "hide folders" and searching for "_h" (see Figure 2). This way you can see the names of each header file and double-click on one to see what functions are inside. Because they also contain implementation, studying them should help you



Palette Window - Script

_h

☐ Only show resources checked out to me
☑ Hide folders

2da_constants_h
2da_data_h
ability_h
ability_summon_h
achievement_core_h
ai_ballista_h
ai_behaviors_h
ai_conditions_h
ai_constants_h
ai_main_h_2
ai_threat_h
aoe_effects_h
approval_h

Figure 2: How to search for all available header files

learn how to code.

You've finished this week's tutorial and learnt the basics of 2DA's and object variables. Next week we'll learn how to solve a reasonably complex problem involving an NPC and a conversation.

Scripting Tutorial 4 – Problem Solving

This week you will be learning how to solve a problem with very little help!

The challenge

Using skills from previous tutorials, implement the following:

1. Each time you talk to a custom NPC, it walks to a random waypoint in the current area.
2. Now make it so that the NPC will never walk to the same waypoint as the one they are currently standing at (i.e., avoid the NPC not walking at all).
3. Now make it so that the NPC will no longer walk to a new waypoint after a particular number of times (e.g., 3) and will instead tell you that it is too tired to walk any further.
4. To possibly make things harder, make it so you only have to change the set of possible waypoints the NPC will walk to by adding/removing the waypoints and **changing only one number in your scripts**. Although this might not make you change your implementation, it's an example of requiring that your code is more flexible and useful.
5. To make things even harder, make it so that once the NPC has walked to a particular number of waypoints, it will not tell you it is too tired but will instead **list all the waypoints that it walked to**.

A solution will be published in the next tutorial!

Scripting Tutorial 5 – Problem Solving Solution and M2DA's

This week you will be learning one method for solving the problem in tutorial 4 as well as using M2DA's to

Solution to the challenge

The srcipt and steps that solve most of the "creature talking/walking" problem from tutorial 4.

1. Scripts for parts 1, 2, 4 and 5:

```
#include "move_random_wp_h"

void main()
{
    object oPC = GetPartyLeader();
    int nWp = MoveCreature("fred", "walk_point", 5);

    DisplayFloatyMessage(oPC, "walking to waypoint #" + IntTo-
String(nWp));
}
```

Snippet 1: Conversation action script to be placed on dialogue line that triggers the NPC to move.

```
#include "move_random_wp_h"

int StartingConditional()
{
    return MovedEnough("fred", 3);
}
```

Snippet 2: Conversation condition script to be placed on start of a dialogue chain "below" snippet 1.

```
#include "move_random_wp_h"

void main()
{
    string sNpcTag = "fred";
    object oNpc = GetObjectByTag(sNpcTag);

    DisplayFloatyMessage(oNpc, "I walked to " + GetWpList(sNpcTag));
}
```

Snippet 3: Conversation action script to be placed at end of same dialogue chain as snippet 2.

```
//:://////////////////////////////////////////////
//:: Helper functions for telling a creature to move to random waypoints.
//:://////////////////////////////////////////////
//:: Created By: Stephen Karpinskyj
//:: Created On: April 9th, 2011
//:://////////////////////////////////////////////

const string VAR_WALK_COUNT = "CREATURE_COUNTER_1";
const string VAR_PREV_WP = "CREATURE_COUNTER_2";
const string VAR_WAYPOINT_LIST = "SURR_PLOT_NAME";

int MoveCreature(string sCreatureTag, string sWpSetTag, int nWpSetSize)
{
    object oCreature = GetObjectByTag(sCreatureTag);
    int nPrevWp = GetLocalInt(oCreature, VAR_PREV_WP);
    int nWalkCount = GetLocalInt(oCreature, VAR_WALK_COUNT);
    string sWpList = GetLocalString(oCreature, VAR_WAYPOINT_LIST);

    int nWp;

    while (nWp == nPrevWp)
        nWp = Random(nWpSetSize);

    object oWp = GetObjectByTag(sWpSetTag, nWp);
    command cMove = CommandMoveToObject(oWp);

    if (nWalkCount == 0)
        sWpList = "";

    AddCommand(oCreature, cMove);

    SetLocalInt(oCreature, VAR_WALK_COUNT, nWalkCount + 1);
    SetLocalInt(oCreature, VAR_PREV_WP, nWp);

    sWpList += " " + GetTag(oWp) + IntToString(nWp);
    SetLocalString(oCreature, VAR_WAYPOINT_LIST, sWpList);

    return nWp;
}

int MovedEnough(string sCreatureTag, int nMax)
{
    object oCreature = GetObjectByTag(sCreatureTag);
    int nWalkCount = GetLocalInt(oCreature, VAR_WALK_COUNT);

    if (nWalkCount == nMax)
    {
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}

string GetWpList(string sCreatureTag)
{
    object oCreature = GetObjectByTag(sCreatureTag);

    return GetLocalString(oCreature, VAR_WAYPOINT_LIST);
}
```

Snippet 4: Include script containing constants and implementation of functions used in above scripts.

2. Other than writing the above scripts and adding them to a conversation attached to your NPC in your area, you should add a set of waypoints (all with tag "create_point") to your area.

The solution above is reuseable because we only need to write the important code once and pass it different parameters (e.g., creature and waypoint tags). While this example might not have called for a reusable solution, (i.e., the code might only ever be used once) it at least demonstratess how you might write re-usable code using parameters and functions in an external file.

The solution to part 5 is not very "nice" because the toolset doesn't allow talk table strings (that includes exported dialogue conversation lines) to be modified at run-time (i.e., a script cannot modify/set a conversation line, only read/get).

Adding a spell

Now that you've got a lot of the basic DA:O modding knowledge, you should be able to start following other tutorials that require basic knowledge. For example, try following this "new spell tutorial" (link).

The tutorial mentions that similar steps are required to add or modify talents and abilities as well.

Although the spell tutorial doesn't mention "M2DA's", it actually asks you to add to one by creating a new 2DA ("ABI_Base_fh"). Because your new 2DA has a special prefix ("ABI_"), the game actually combines your new "ABI_Base_fh" file with all other 2DA files that have the "ABI_" prefix.

M2DA's are useful because you can add to existing 2DA's (e.g., a list of "placeable_types") without copying potentially hundreds of rows from the original 2DA (e.g., you can create a new 2DA called "placeable_types_something" with your new placeables and the game will merge this with other "placeable_types_*" 2DA's to form a "placeable_types" M2DA).

For a list of M2DA's, see the "M2DA_base" worksheet in the "2DA_base" 2DA file.

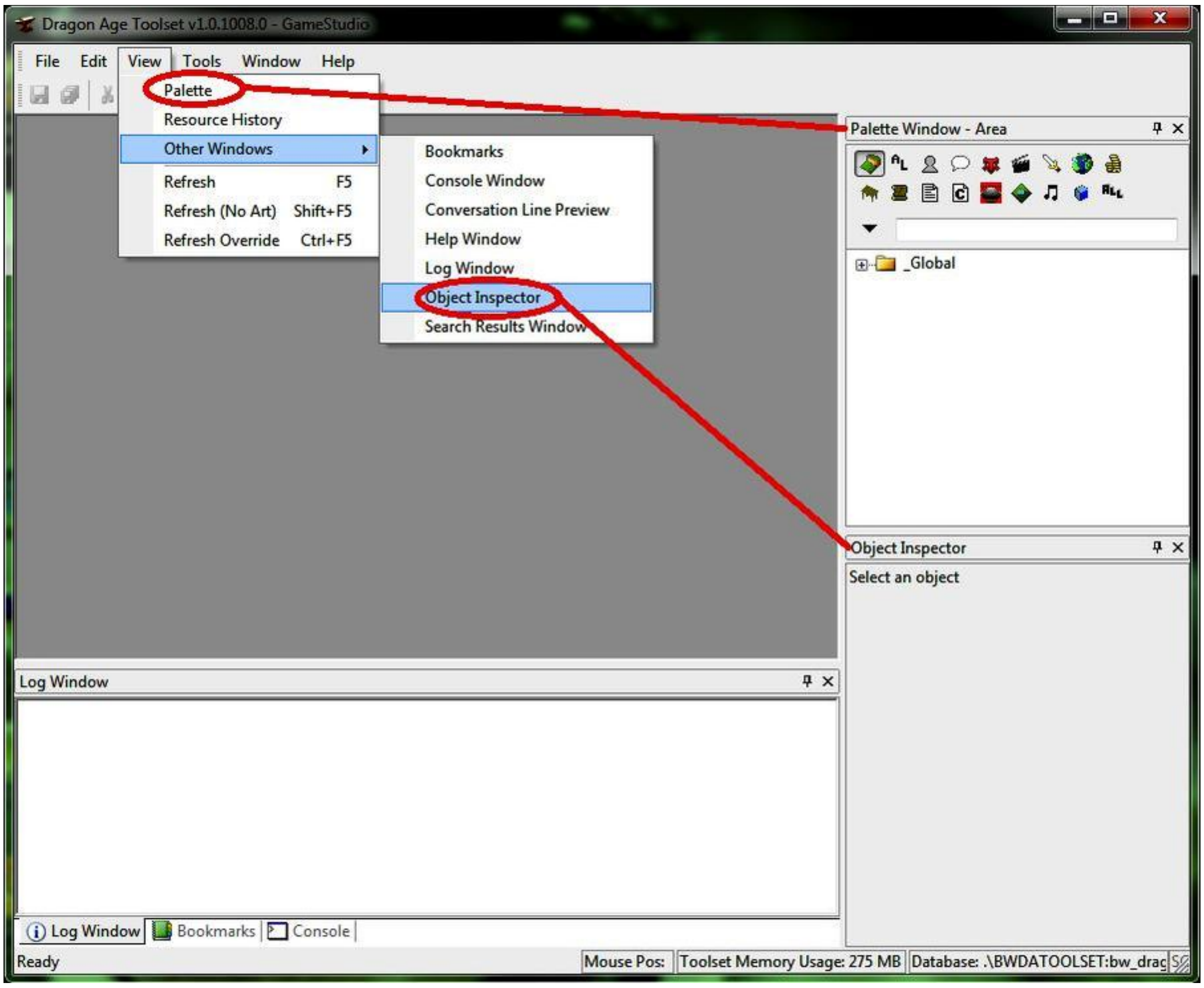
Interestingly, you can create new M2DA's using this method because "M2DA_base" is itself an M2DA (e.g., create a new 2DA called "M2DA_something" that has a list of your new M2DA prefixes). See the builder wiki for more info (link**).**

ART & DESIGN

Tutorial 1 - Introduction

The Toolset

When you first open the toolset it looks pretty empty, the first thing you will want to do is make sure you have the Palette and Object inspector windows open.



It is also a good idea to have the Log window open, so you can see what is happening this is particularly useful when you are building, rendering in and exporting levels.

Note: If you have mods or expansions installed make sure you do not use the resources from them in your assignment. We mark on computers that have the same set up as the ones at uni which do not have the various expansions or extra mods installed, therefore they will not have the resources and they will not appear in your mod when we play it, often this will cause it to crash.

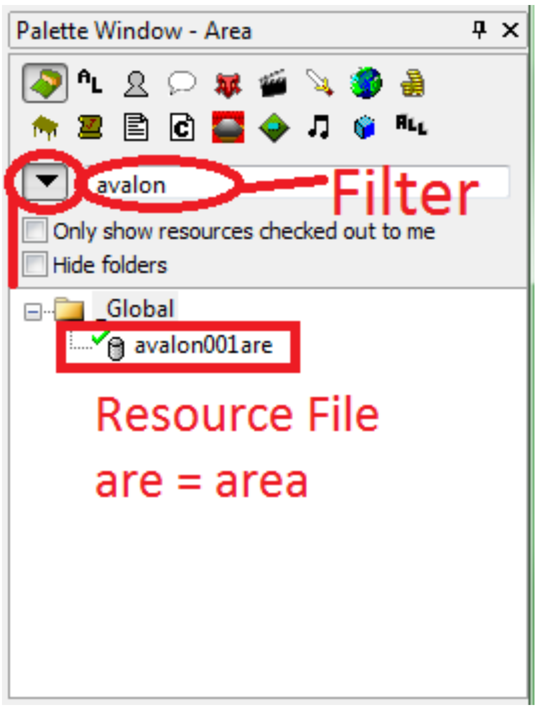
The main menu contains all the standard file, edit, tool, window and help properties and are consistent across all resource editing windows, I recommend you become familiar with them.

The top menu contains editing type specific controls, it contains most of your basic functions like save etc as well as all functions related to resource manipulation, rendering, building and much more. These menu bars are Resource specific so they will change with the type of resource you are editing.

The palette window is where you find all the game resources such as placeables, items, character models, interior terrain tiles etc. Basically every single resource in the DA:O game and any mods or expansions you have installed.

Each resource is organised into a category you have:

Areas
Area List
Characters
Conversations
Creatures
Cut Scenes
Items
Maps
Merchants
Placeables
Plots
Scripts
Client Scripts
Stages
Triggers
Sounds
Models
Display All



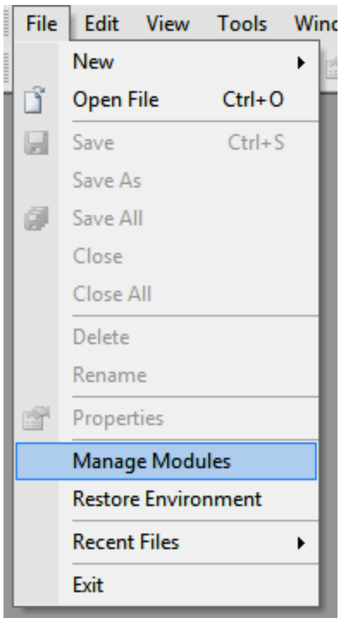
You also have a filter option where you can search for resources, this is mainly useful when you have the Hide folders option checked which is accessed by clicking on the triangle next to the filter box.

As you will notice the resources themselves, in this views do not have very descriptive icons, to see what each of the models looks like you can look up the Lists with screen shots on the builder Wiki:

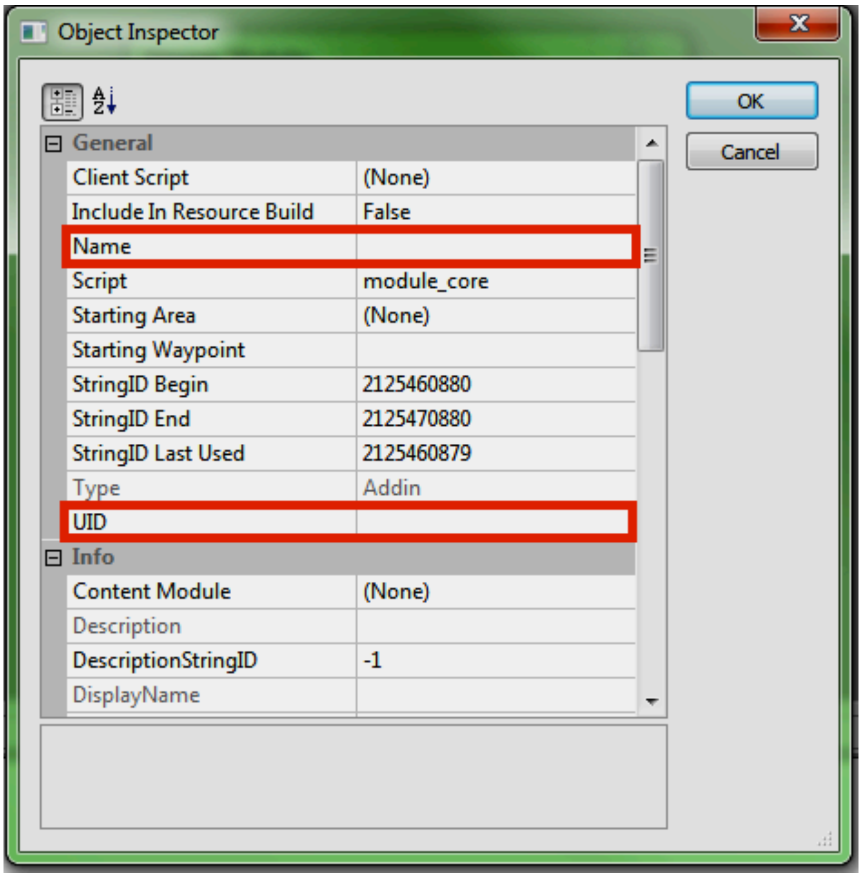
- Models
  http://social.bioware.com/wiki/datoolset/index.php/Model_list
- Vegetation
  http://social.bioware.com/wiki/datoolset/index.php/Vegetation_list
- Terrain Textures
  http://social.bioware.com/wiki/datoolset/index.php/Terrain_list
- Vista Objects
  http://social.bioware.com/wiki/datoolset/index.php/Vista_object

The object inspector is where you will find all the resource properties, but we will look at that in more detail later.

First thing we are going to do is create a module, Stephen will go into this in more detail next but for the moment we will simply do a basic set up. Go to **File > Manage Modules**.



This will open up the modules manager, first thing we are going to do is **click New...** This will open a new module properties window. You have a huge number of options here and most of them you will want to tweak later on when you have some basic bits set up, but for the time being we will just set up the two most important things, the Name and UID fields. ***Name*** is the visible name of the Module In this case put in ***Game Studio Tutorials***. The ***UID*** is the Mod's unique identifier string in this case put ***GS1TUTS***. Then Click OK You will now have a new module called **Game Studio Tutorial**s Select it and **click Open**.



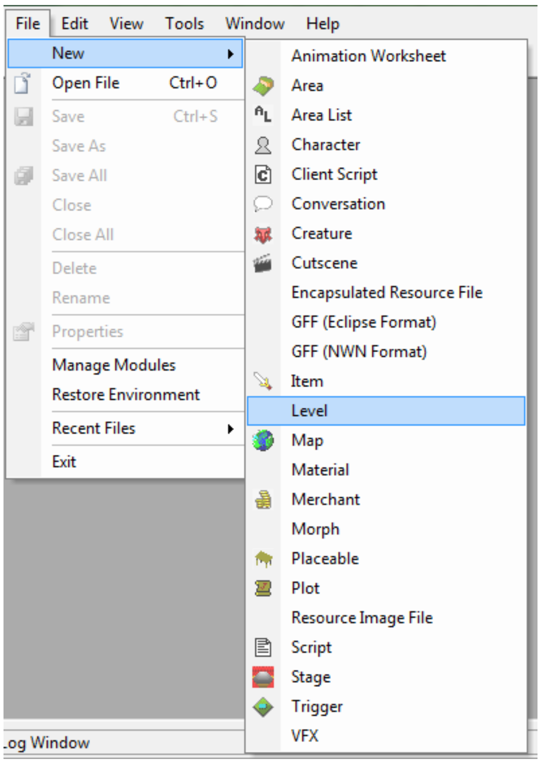
General	
Client Script	(None)
Include In Resource Build	False
Name	
Script	module_core
Starting Area	(None)
Starting Waypoint	
StringID Begin	2125460880
StringID End	2125470880
StringID Last Used	2125460879
Type	Addin
UID	
Info	
Content Module	(None)
Description	
DescriptionStringID	-1
DisplayName	

You now have a new module, however there is nothing in it. If you want to find the mod files on the computer you will find the them in the BioWare Folder > AddIns > GS1TUT (Or whatever UID you choose to use for your mod).

If you want to see a breakdown of the module properties see:
http://social.bioware.com/wiki/datoolset/index.php/Module#Properties

Levels 101

In the Original NWN2 toolset 'Areas' and 'Levels' were synonymous however with the new DA:O resource structure system, Levels and areas are two different resources. **Levels are the static parts of an area** e.g. the terrain itself, buildings, trees, terrain textures, placeables and junk that the player doesn't interact with aka the decoration. Levels can be interior or exterior and contain quite an array of complex elements. Once a level is finalised it is imported to an area so that the designers can build upon the static terrain and create the dynamic elements. It is important to remember the spaces needed for the dynamic elements when you are building a level, you may choose to use place holder objects while you build a level just so you make sure to leave enough space. When it is finished the level is exported, this exported file includes, lightmaps, water reflections, placeables, trees and grass, walk maps, sounds etc . These are then imported into an area masking the framework of it.



To create a new level you need to go to **File > New > Level**. Once this is done select **Room Level > Next > Finished**. Now you have an empty room with no floor, walls or ceiling. First things first you will need to learn to navigate in this space, There are two styles of camera controls Fly cam and 3DSMax, you can change the control system by clicking on the camera control buttons . The Alt is the 3DSMax system and the one I prefer:

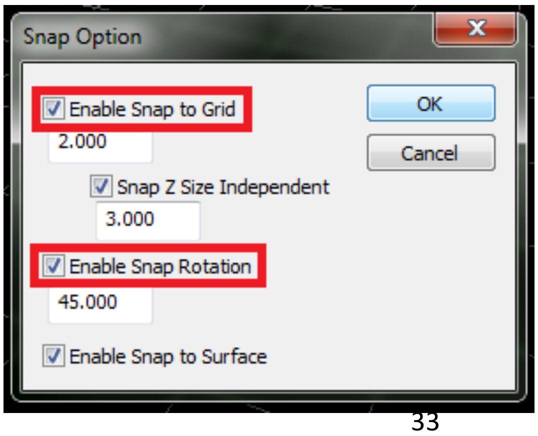
- **Press and hold middle mouse button = pan**
- **Press and hold middle mouse button + Alt Key = Rotate**
- **Mouse wheel = zoom**
- **Ctrl + mouse wheel = Speed Zoom**
- **Z = centre and zoom in on the centre of the level**

But you can also to the fly cam (WASD)

- **A = Pan Left**
- **D = Pan right**
- **W = Zoom in**
- **S = Zoom Out**
- **Hold Centre Mouse Button = Rotate**
- **scroll wheel = Zoom**
- **Ctrl + hold centre mouse button = rotate on light on sight**
- **Z = centre and zoom in on the centre of the level**

The next thing you will notice is that a new window had appeared down the left side I call this the resource tree as it displays a tree of all the elements or resources within the resource you are editing.

Once you have this started we will start to place a floor. One thing you will want to do before building interiors at least for the floor and walls is turn on Grid Snapping. Click on the little magnate on the top the tool bar . This will bring up the snap tool bar ***Enable Snap to Grid at 2.000*** and ***Enable Snap Rotation at 45.000***, This will just make the basic room setting up easier.


Snap Option

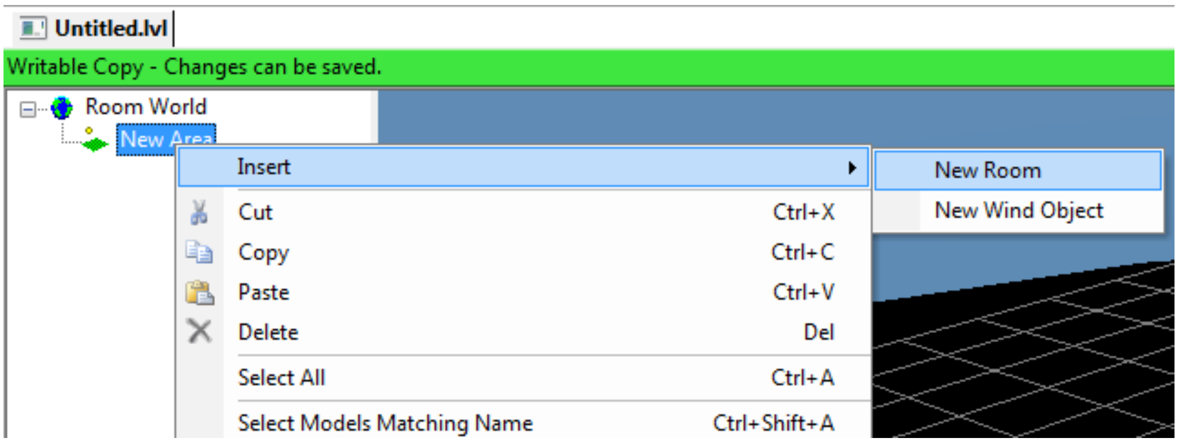
☑ Enable Snap to Grid
2.000
☑ Snap Z Size Independent
3.000
☑ Enable Snap Rotation
45.000
☑ Enable Snap to Surface

OK
Cancel

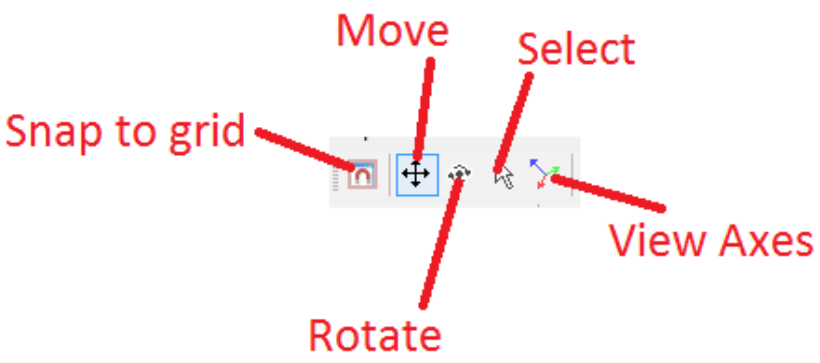
There are a couple of things you will need to do before you start placing room tiles, first make sure the 'View Models Fully Lit' button ( )is on this means that it will light everything equally in the level and have no light mapping, and since we have no lights in the level currently this is what we need to see what we are doing.

Next click on the New Area Label in the Resource Tree, This needs to be selected to place your floor or any models in the Level. You will also need to set up a few properties for the new area before we begin. Once you have selected 'New Area' look at the Object Inspector, there are a whole bunch of properties here which you can change and tweak to fit your level, the two that you **MUST CHANGE** or you won't be able to build you level is the Layout Name and Name. The Layout Name is the name of exported layout file it has to be less than 8 characters with no spaces or symbols. I will set this one to ***Layout Name: GSlvl1*** and set the ***Name: gs1_interior_01***.

Now right click on the 'New Area' and select **Insert > New Room**. This will Create a 'New Room' child node under the parent 'New Area'. You should now begin to see that the DA:O System is very hierarchical. You have parent objects or nodes and underneath them you have child nodes, one parent node can have many children but a child may only have one parent.

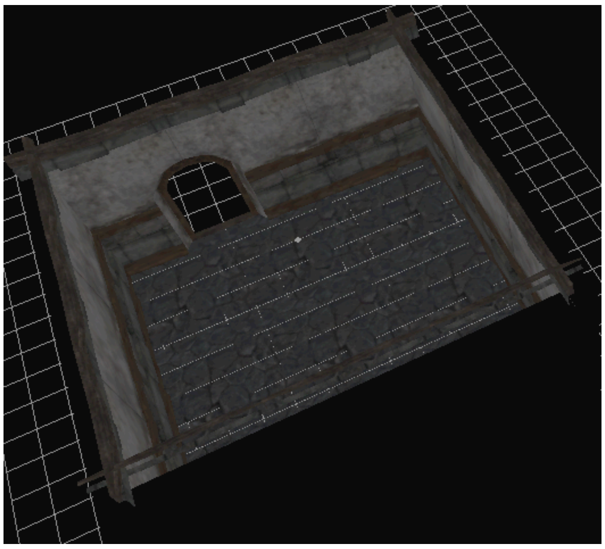


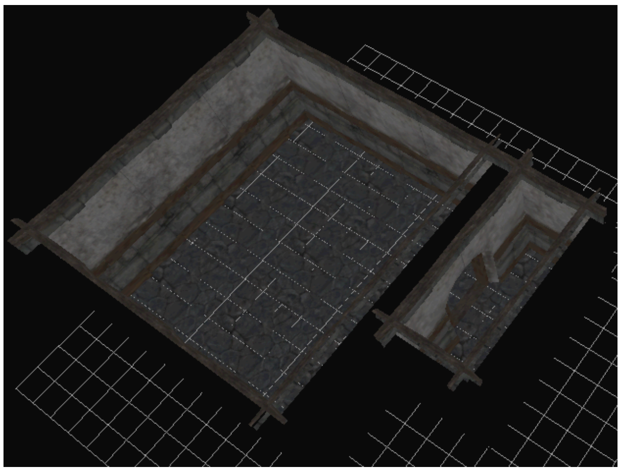
Once you have got all this set up it is important to mention the Move, Rotate, Select and view axes tools, which you will definitely need in the next step. These can be found to the right in the top menu, they allow you to manipulate the objects or tiles within the level.



Next comes the fun part, select a floor! To do this click on the models button in the Palette window (The blue box) and type floor into the filter, check hide folders if you want to see the individual resources. Once you have decided on the floor you want, I will use **fca_floor01_0**. Select the New Room in the Resource Tree and place the tile of floor on the grid by left clicking. You can now Copy and paste this tile using **Ctrl + C** and **Ctrl + V** to duplicate the tiles. Use **Ctrl + left click to select multiple tiles** if you want to duplicate multiple tiles. Copy and paste the original tile, use the Move tool and select the axis you wish to move the tile along and arrange them next to one another. continue this process until the floor is as big as the room you wish to create. For the moment keep it square or rectangular.

Next comes the Walls. A good trick to remember is the first three letters of the file name of the element you used is the folder name and within the folder they often have matching floor, wall and ceiling tiles. I am going to remove the filter and get back the folders then expand the **'fca' folder > wall01 > fca_wall01_0**. Now I have a wall tile I can begin to build with, Place the wall at the edge of the floor tile and place it. Copy and paste it till you have a whole wall, when you get to a corner instead of simply moving the tile rotate it as well (this is where the snap to rotate helps) place it at the corner and continue until to have a whole room, you can now add the ceiling the same way. If you have had experience with Never Winter Nights you will know there was no ceiling, however in DA:O you need one as the players fixed camera angle is low enough that they can actually see the ceiling.

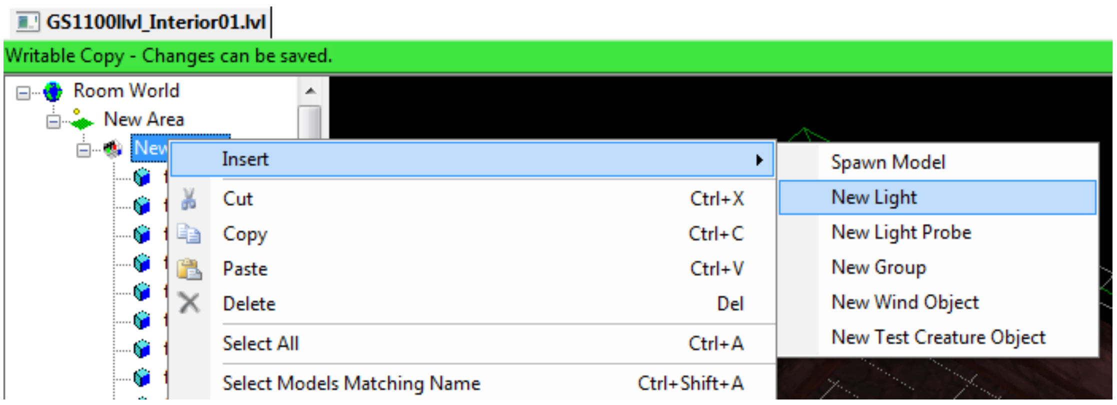




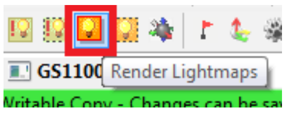
If you are looking at adding a door way between a corridor or rooms you will need to find the relevant tile, not only that but you will make sure you put a matching one on the back **as all these walls are one sided**. You will lose marks in your assignment if we find one sided walls in your interior. It is also a good idea **<u>to add a New Room to the resource Tree and place the new room's tiles within that 'New Room'</u>**, it will give you allot more flexibility later and you can also name the rooms using the Name property which helps with ease or reference later

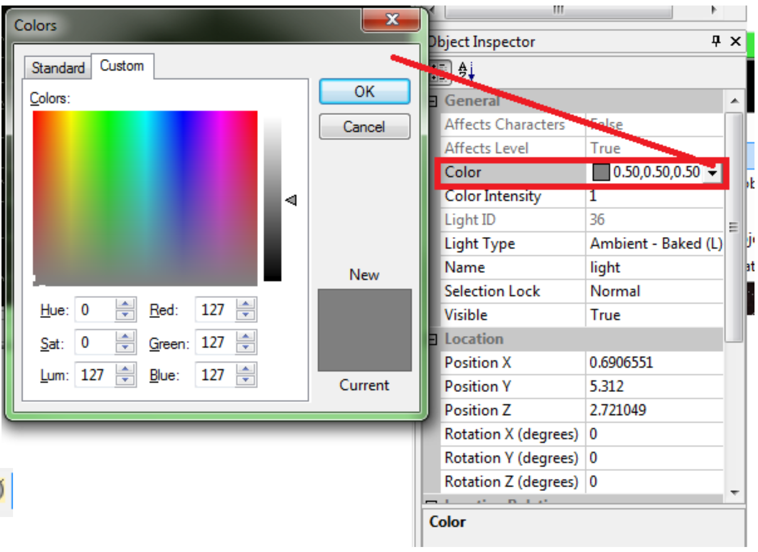
I will place the tile form **fca > door01 >fca_door01_0** in the space of one of my walls, you may need to juggle pieces around a bit and turn your snap to 1 as the door walls can be differently sized. Now you will need to copy and paste that same doorway and rotate it 180 degrees. Put it on the other side of the door way so it matches up and now you can build another small room. You will add the doors in the Area editor, as they are considered dynamic elements.

The next thing we want to do is add two lights, so right click on the New Room label in the Resource Tree and click **Insert > New Light.**



Once you have done this place select it, then go to the Object inspector. This is where you have all the possible light settings, including colour type intensity etc. For the moment just *set **Light Type: Point - Static(lc)***, you will also want to change the ***Name to PointStatic01***. If you want to can fiddle with the colour and intensity, but at the moment I will leave it how it is. Select it and move it to where you want, you may want to turn off the grid snapping when you want to position these. Realistically you will need one point static light per 'room' so you don't end up with very dark rooms. I am just going to add a light to the other room Set it to Point Static and call it PointStatic02. Now I will add another light to the main give it a ***Name of Ambient01*** and set it to ***Light Type: Ambient Baked(L)*** You will want to set the colour on this one to something dark, this becomes the colour/tint of the shadows. To do This go to Color, select it and click on the down arrow. Here you can select your colour. Then click OK. Once that is done click on the render light maps button, watch the log it will tell you when it was successful. I will talk more about these nest week but for the, moment we will just render it. To see the light map turn off the global lighting and turn on light map (LM).

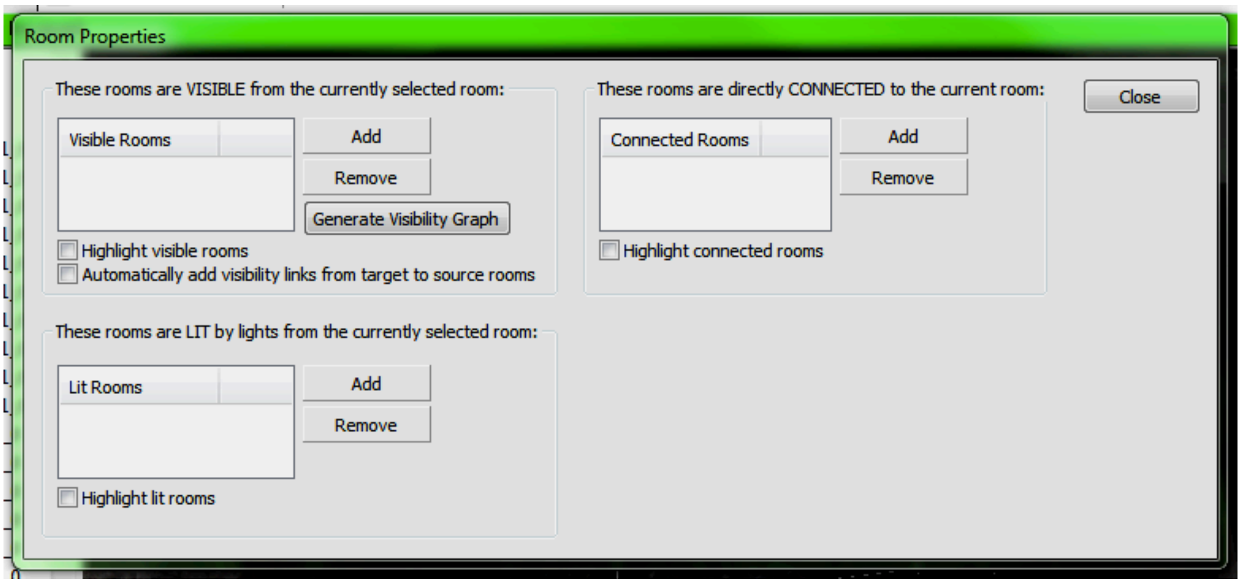




Hue:	0
Sat:	0
Lum:	127
Red:	127
Green:	127
Blue:	127

Affects Characters	False
Affects Level	True
Color	0.50,0.50,0.50
Color Intensity	1
Light ID	36
Light Type	Ambient - Baked (L)
Name	light
Selection Lock	Normal
Visible	True
Position X	0.6906551
Position Y	5.312
Position Z	2.721049
Rotation X (degrees)	0
Rotation Y (degrees)	0
Rotation Z (degrees)	0

Once you have built the two rooms you will need to connect the rooms using the "Room Properties tool". make sure they are uniquely named by selecting the room then changing its name in the Object Inspector. Select the room in the resource tree and click on the "Room properties button". Here you will see a window like this:



These rooms are VISIBLE from the currently selected room:

Visible Rooms

Highlight visible rooms

Automatically add visibility links from target to source rooms

These rooms are directly CONNECTED to the current room:

Connected Rooms

Highlight connected rooms

These rooms are LIT by lights from the currently selected room:

Lit Rooms

Highlight lit rooms

Once you are here you can control how the room rendering using the VISIBLE ROOMS section, you can set-up which room is connected to which using the CONNECTED ROOMS section and you can also set up the room lighting, what lights from what rooms effect other rooms.

Now you have a level, it is not quite ready for importing into an area but it is a good start. next week we will look at External Levels and more on lights as well as area props.

Now save your level remembering the DA:O Naming Conventions, e.g. **GS1100lvl_Interior01.lvl**.

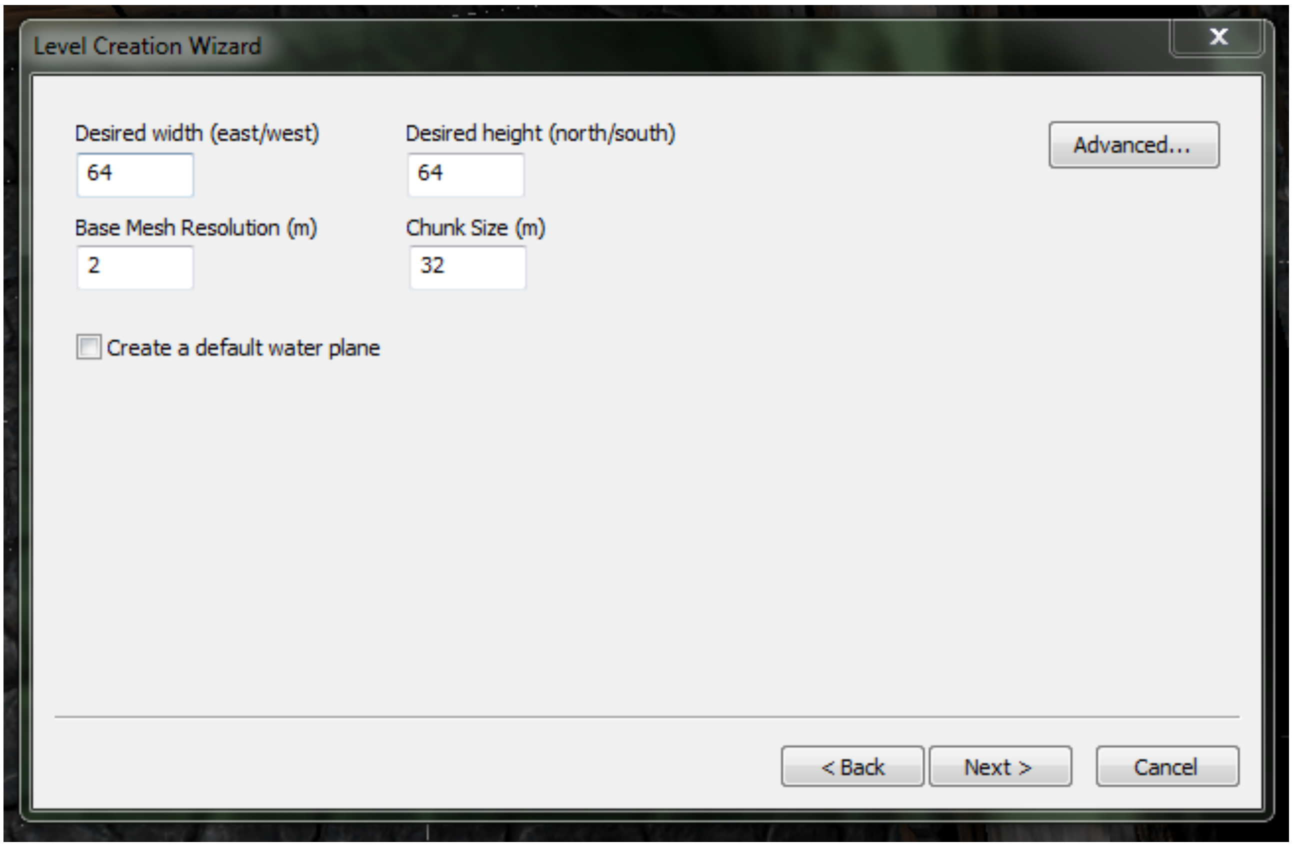
NOTE: the .lvl file itself can be saved anywhere it is the exported files (which we will get to next week) that are important you can take home the .lvl file without editing any databases as it is considered an art resource, however when you start getting to exported levels and areas you will have to start exporting and importing the game objects properly.

This is a start, but not a playable area or level yet. Today once you have formed your groups I want you to start exploring the interior tiles, look through the toolset at all the floor, walls and ceilings and start making notes about what tiles suit what themes, tile sizes etc. You will find this information very useful when you come to building your Levels for your mod.

Tutorial 2 – Exteriors, Water, Vegetation and Models

Exteriors

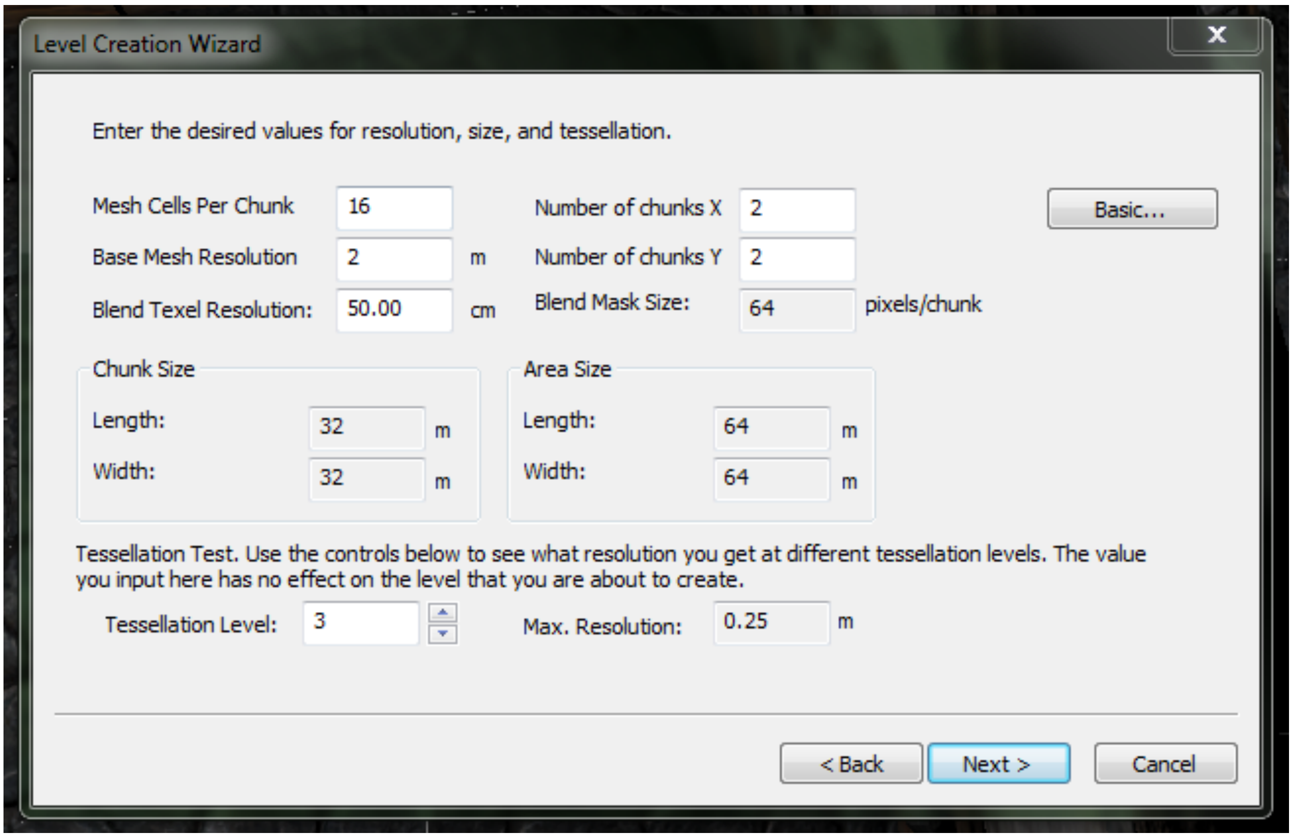
So Last week we created an interior today we will be looking at exteriors. To do that we will first make sure our module is selected, to check go to **File > Manage modules** and make sure that the module you wish to be working on is open. Next go to **File > New > Level**. This will bring up a new window, **keep Terrain (Landscape) Level selected**, then click next you will then see this window:



Desired width (east/west): 64
Desired height (north/south): 64
Base Mesh Resolution (m): 2
Chunk Size (m): 32
☐ Create a default water plane

- Desired width (east/west) & Desired height (north/south) - is the terrain dimensions in meters, 64 by 64 is about the minimum that you'd want to use and 256 by 256 is probably the largest we'd recommend for performance sake.
- Base mesh resolution - the size of the individual triangles that make up the terrain mesh, we'll talk more on this later when we get to the tessellation tool.
- Chunk size - chunks are the basic simulation unit used by the engine, the default 32m is a good size it is best to leave it as is.
- Create a default water plane - if you know you really want to add a water plane then you can select it here, but generally I wouldn't bother as it is very easy to add later.

For the moment leave these how they are and click on the "Advanced..." button. This changes the display to the advanced options.



Level Creation Wizard

Enter the desired values for resolution, size, and tessellation.

Mesh Cells Per Chunk	16		Number of chunks X	2		Basic...
Base Mesh Resolution	2	m	Number of chunks Y	2		
Blend Texel Resolution:	50.00	cm	Blend Mask Size:	64	pixels/chunk	

Chunk Size

Length:	32	m
Width:	32	m

Area Size

Length:	64	m
Width:	64	m

Tessellation Test. Use the controls below to see what resolution you get at different tessellation levels. The value you input here has no effect on the level that you are about to create.

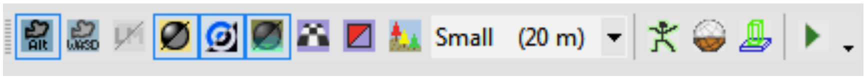
Tessellation Level: 3 Max. Resolution: 0.25 m

< Back Next > Cancel

Here you can do some fine tweaking. Really the only thing you may want to tweak here is the Tessellation level, but for the moment we will just leave it all as it and click back to the basic view using the "Basic..." button. Now click "Next>". This is just a summary of your terrain if you decide you want it bigger or want to tweak something, you can just click "< Back" and edit your terrain. We'll just click "Next >" and get to work.

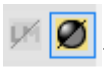
There are a few import5ant menus that I want to go through now, so that when we get to editing you'll know what's what.

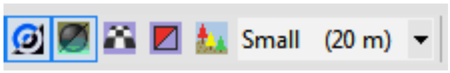
View Controls



This menu controls what control system you have mapped as well as what you can see in your level and how you see it.

These two buttons control the style of camera controls you wish to use (as discussed last week)

These two buttons control the lighting style you will see, the LM (Light Mapping) one is currently greyed out because we haven't generated a light map. The second button is universal or flat lighting, it just means that everything is equally lit, this is good for the initial building stage. If you can only see black when you first make a level make sure this is set on so that you can see what you are doing.

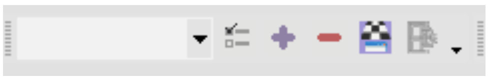


The buttons controls what you see in your view e.g. Buffer effects, Atmosphere effects (such as fog), Terrain chunk boundaries, Highlighting of accessible and inaccessible terrain, Tree draw distance settings and toolset render distance. The toolset render distance is by default set very small (20m) you will probably want to increase it, the amount of vegetation you have and the power of your computer will control what render distance you want but if you have a good enough machine you should be ok to put it to the Very Far setting (300m).

These buttons control the view preferences for objects, e.g. tactical rim lighting, fade cut away and object collision boundaries. At this stage, while you build your level, will just be using the last two.

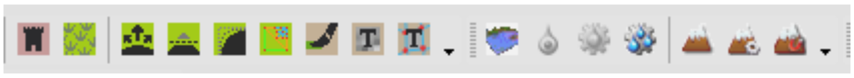
This button enables or disables the refreshing of the render window. I haven't used this yet, but you may find it useful at some stage.

Area Controls



At the moment these are mostly greyed out, because your level does not yet have its exportable settings set up. I will show you that later today however, it is a good idea to be familiar with them. The drop down menu is the list of areas that the level can be exported to, generally you will only have one, however if you wanted a dusk or night version of the same level you can create a second and adjust the lighting and environmental effects(fog etc.) to give you all the necessary rendered light mapping. The second is your area properties, you will use this allot later when we get to exporting and setting up some of our area settings. The "+" button is to create a new exportable area, which does just what it says, its partner the "-" button does the opposite it deletes an exportable area. The fifth grid looking button is the "Chunk management tool" it allows you to tweak the setting of each chunk. The final is the Room Properties tool (see last week's tutorial).

Terrain and Water Editing Tools



This is the main menu you will use when constructing your terrain level:

This is the model placement button, it allows you to place models in the level.

This is the Scatter Object tool, it allows you to place vegetation e.g. trees and grass in your level.

These are the terrain editing tools: 1. Deform (Raise & lower), 2.Plateau, 3. Smooth, 4. Tessellate

These are the terrain texturing tools, that allow you to paint textures on the terrain: 1. Texture Paint, 2. Texture Smooth (Like a blend using transparency), 3. Relax map (if a texture is stretched it will unstretch it).

For a really good break down of the terrain tools and texture tools inc. their properties see the wiki : http://social.bioware.com/wiki/datoolset/index.php/Terrain_mesh

These are the water collision tools, most of the properties you will set for water will be in the water's properties or the other options, Surface Colour & Wave Settings, that you will find in the object

inspector. However if you want to edit a water planes collision style etc you should look here.
1. Automatically Tessellate the water to allow for increased vertices at the shoreline, 2. Show Water collision Mesh, 3. Generate water collision geometry for the selected water node, 3. Generate water collision geometry for all water nodes.

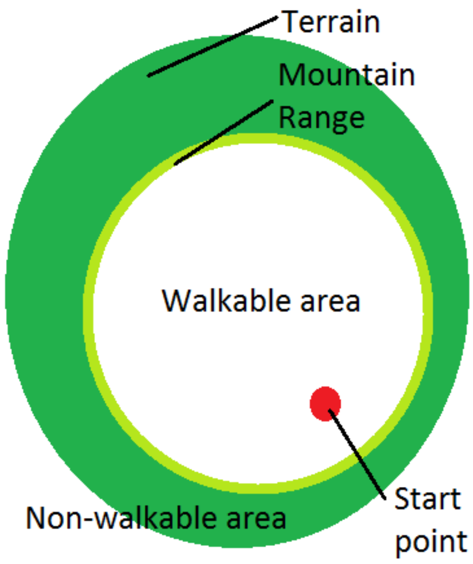
These are the terrain collision tools. 1. Show Terrain Collision, 2. Build Terrain Collision, 3. Snap Terrain Collision.

Light Map Rendering

These are the light map rendering tools. Once your level is set up as an area and has some Lighting set up you can export the light maps. The options are 1. Render Light maps (Cached Scene), 2. Render Light maps for selected room or chunk (Cached Scene), **3. Render Light maps, 4. Render Light maps for selected room or chunk, 5. Render Light Probes**. I mainly use the last 3, particularly 3 & 5. We will deal with light probes later but what they do is generate a water's reflection relative to the level, e.g. trees/buildings overhanging the water get reflected.

Start Point and Walk Map Rendering

These tools allow you to set up the level walk maps. 1. Set up Start Point, 2. Toggle display of path finding nodes, 3. Generate path finding for active area, 4. Generate path finding but skip the footstep-sound generation step. You always need to place a start point to generate a walk map, it sets the area within which the player will move. e.g. If you had an level which was surrounded my an impassable mountain range, you wouldn't want the player to walk on the other side of the range. So you set the start place inside the range and the path finding generates, in relation to that point, where the player can and can't go. You can then tweak the automatically generated path finding with the Collision tools discussed earlier.



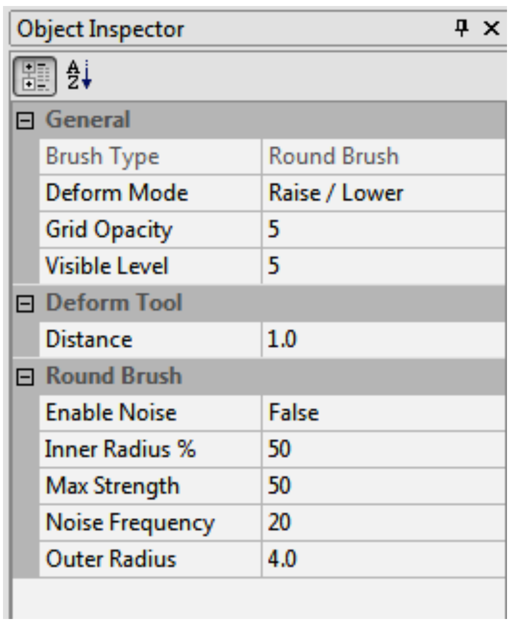
Level Exporting

This is where the bugs pop up! These are the tools you use to export your level ready for use in an area. 1. Post Selection to Local, 2.Post trees to Local, 3. Post only object Locations to Local, 4. Post to Local**, 5. Do all local posts**. Generally it is safest to do 5, which can take a while but exports everything in case you forgot do something, or a change you made inadvertently changed something else, it renders the light maps, walk-meshes, exports the trees, and object placements as well as the terrain mesh.

Editing Terrain

Now we have our head around what tools we have available we can start editing.

Select the Deform tool. This deforms the terrain **left click to raises the terrain** and **right click to lower**. When you have the tool selected look at the Object Inspector. Here you have a range of brush options:



General	
Brush Type	Round Brush
Deform Mode	Raise / Lower
Grid Opacity	5
Visible Level	5
Deform Tool	
Distance	1.0
Round Brush	
Enable Noise	False
Inner Radius %	50
Max Strength	50
Noise Frequency	20
Outer Radius	4.0

These options allow you to tweak the brush size, pressure, noise, deform mode etc. Here you can set the inner radius (%) and outer radius (brush size), the inner radius is the area of highest intensity; the outer radius is the area over which the pressure drops off. You can then set the max strength which equates to the pressure or how forceful the terrain moulding will be.

If you are struggling to see the grid, you can adjust the grid opacity, this is especially useful when you start looking at using the tessellation brush.

You can also set the mode of the brush, most of the time you will only need to raise or lower the terrain but sometimes you might want to have very angular odd shaped terrain then you could use the Extrude along normals mode. Using the left lick or right click you can extrude or depress along the normals of the plane, experiment to see what I mean, it works most effectively on hills or already deformed terrain.

The Platteau tool flattens the terrain to the same height as the place you initially clicked. This is useful in resetting the terrain to flat or creating plateaus for buildings etc. **NOTE if you have extruded along normals your terrain grid will be deformed so just be wary of using in your level it unless you are sure.**

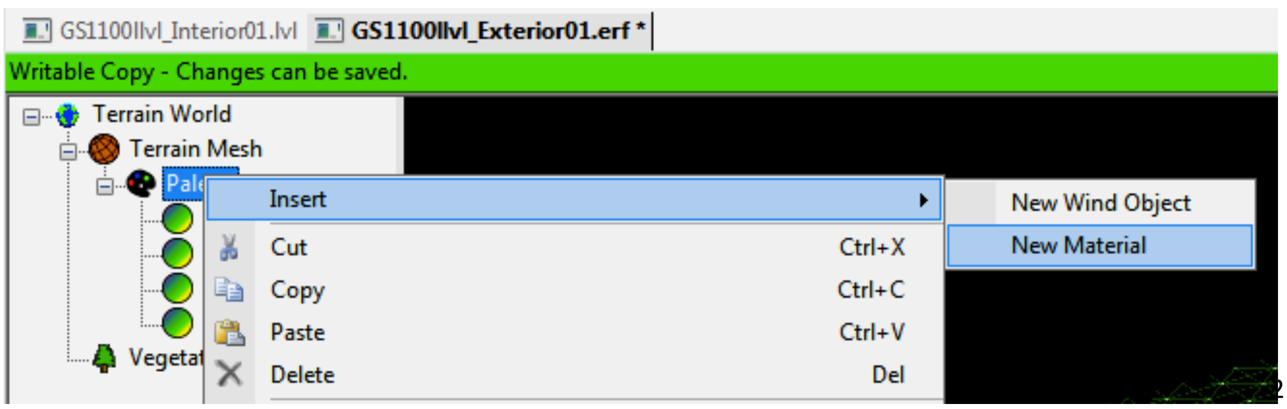
NOTE : You can always use **ctrl + z to UNDO** anything you have done

The Smooth tool helps you to smooth rough slopes, it is best used in conjunction with the tessellation tool.

The Tessellation Tool, allows you change the resolution of the terrain's mesh, increasing or decreasing the number of triangles a given region, **Left Click to tessellate** and **Right Click to un-tessellate**.

Terrain Texturing

When you want to texture your terrain it is a good idea to work out what textures you want to use, check out the wiki for a list with pictures (http://social.bioware.com/wiki/datoolset/index.php/Terrain_list). Pic the texture you want, I will use **ter_swroots_d**. Select the Palette in the Resource Tree, right click > Insert > New Material.



Once you have this you will have a resource called New Material in the Resource Tree. Select it and go to the Object Inspector. Set the Name to something descriptive, I will call mine TreeRoots. I will Leave the rest of the settings in General alone, but if you wanted to later you could go back and tweak things like the sound associated with the texture. UV Tile is basically the amount of tiling a texture has 8 is generally a good number for terrain textures, but play around the best tiling for the texture depends on what you are looking for. In this case I will set the **UVTile to 8**. Now I will select the diffuse texture and normal map, you can also set a specula map if you want, but for the moment I will leave it at None. Click on the Diffuse box and then on the "..." button to the right you can now look for your texture, I am looking for ter_swroots_d so I will start typing that. A good rule of thumb is that anything beginning with ter is a terrain texture. select **ter_swroots_d**, the d stands for diffuse. then click on normal and select the file of the same name with "**_n**" rather than "**_d**" at the end this is the normal map. Now my texture is ready!! Click on the Texture Paint button and you will see the new texture TreeRoots in the texture selection you can now select it and paint.

Note: if you are looking to make your own terrain textures, check out the Material editor in better detail: http://social.bioware.com/wiki/datoolset/index.php/Material_editor. It looks like the editor allows for TGA textures so once you add them into the Art Resources databases you should be able to use them. If you really want to make your own textures you will have to investigate it yourself, we don't cover it in the tutorials.

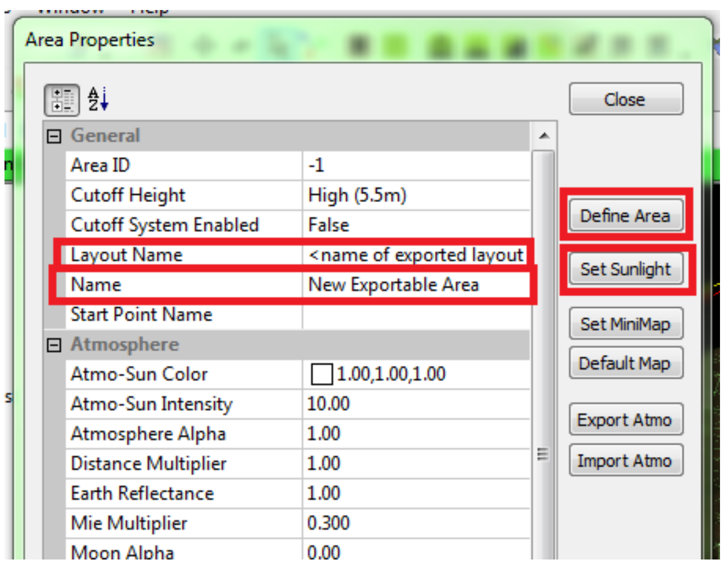
Once you have painted your textures you can tweak the texture settings as well as play with smoothing and relaxing the textures.

Water

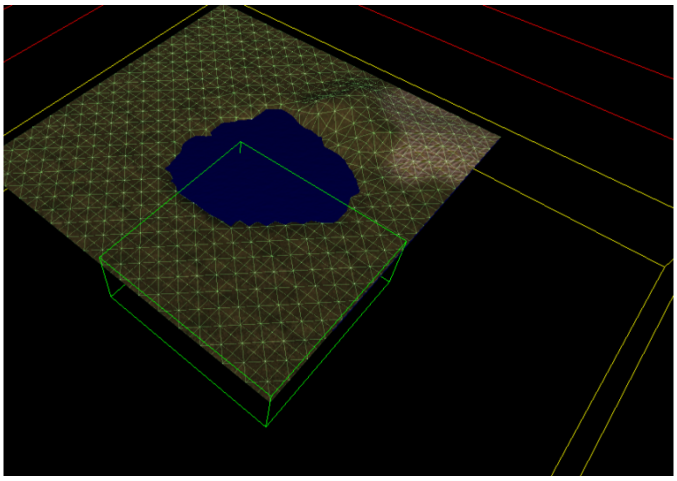
Adding water to a terrain level is simple. Right Click on Terrain World in the Resource Tree and click **Insert > New Water Mesh**. Depending on what you want to do with your water you will need to adjust its heigh and size. We want to make a pond so we will move the water mesh just below our terrain layer (you may need to turn snapping off so it is only just under the terrain. Now we will use the deform tool to create a lake, remember right click, lowers the terrain. Now you have a lake, at the moment it looks a like a black pit, to fix this we will set up the area and it's lighting then come back to the water.

Making an Exportable Area

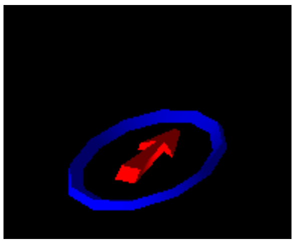
First select the Terrain world in the Resource Tree and then click on the "+" button. This will open up a new Area Properties window. Two things you MUST do before you continue is set: **Layout Name**, this will be the name of the exported level file that you will use in the area, it must be unique and **Name**, this is the human readable name of the Area.



General	
Area ID	-1
Cutoff Height	High (5.5m)
Cutoff System Enabled	False
Layout Name	<name of exported layout
Name	New Exportable Area
Start Point Name	
Atmosphere	
Atmo-Sun Color	☐1.00,1.00,1.00
Atmo-Sun Intensity	10.00
Atmosphere Alpha	1.00
Distance Multiplier	1.00
Earth Reflectance	1.00
Mie Multiplier	0.300
Moon Alpha	0.00



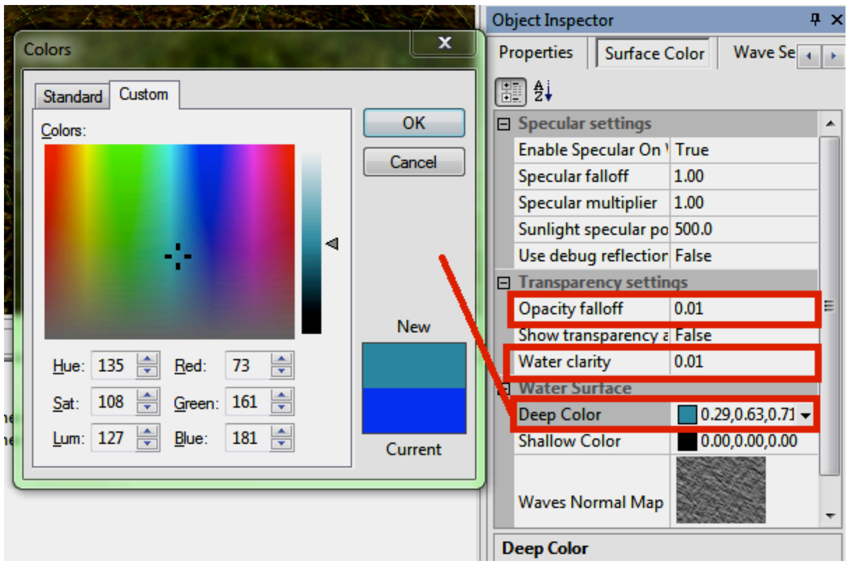
What you will probably also want to do is Define the Area and Set the Sunlight. To Define the Area Click "Define Area". You will see a green square a yellow one and a red one these define the area, green defines the walkable space, you will want to make sure that this covers the whole area of your level that you want the player to walk on to do this left click and drag, the green section will snap to the level chunks. Once you have done that, you will want to set up the sunlight. Click on Set Sunlight the boxes will disappear and a blue circle with a red arrow in it will appear click and drag on the mouse to change the light Sun light's direction.



Click close, now you have a workable area.

Water - Part 2

Now you have your lighting you can see the water. If the water is to large or too small you can select it and then set the X & Y size in the properties inspector under Water Mesh. I'm going to make mine a bit smaller because it is only a pond. When you scale down, one thing you will notice is that the water becomes more transparent, this is because the smaller it is the program assumes that it is shallower water, if you want the water to stay opaque I recommend leaving it a bit larger than you strictly need. You can edit some of the transparency properties in the Surface Colour section , but I have not found them to be that effective for small bodies of water. I'll set it back to 50 x 50 then go to the Surface Colour section here you can change the opacity fall off and the water's colour. The "Surface Color" window can be found in the object Inspector next to Properties and Wave Settings. I am trying to make the lake less transparent so I am going to set the **Opacity falloff to 0.01** and the **Water Clarity to 0.01** I am also going to change the colour of the water to something a little more murkey. I woudl suggest all the settings you can find in this window and in the wave settings, it will allow you to crontol how yor water appears in game much more. It allows you to not only set colour or transparency but wave size, the wtare normal map, the warters specular features etc.



Hue: 135 Red: 73
Sat: 108 Green: 161
Lum: 127 Blue: 181

Specular settings	
Enable Specular On	True
Specular falloff	1.00
Specular multiplier	1.00
Sunlight specular po	500.0
Use debug reflectior	False
Transparency settings	
Opacity falloff	0.01
Show transparency a	False
Water clarity	0.01
Water Surface	
Deep Color	0.29,0.63,0.71
Shallow Color	0.00,0.00,0.00
Waves Normal Map	

Deep Color

Vegitation

Now we will want to add some trees and grass to our level. To do that we will need to add some vegitation, pick which vegitation you wish to add then we will add it, see the wiki for a list (http://social.bioware.com/wiki/datoolset/index.php/Vegetation_list). I am going to use **tre_f_oakhero** and **gra_m_gtall**. To add vegetation to a level, you will first need to right click on the Vegetation heading in the Resource Tree then click **Insert > New Tree Controller** select the **tre_f_oakhero**, then add another and add the **gra_m_gtall**. Now they are added select the scatter brush and you can 'paint' them onto the terrain.

Like the terrain brush you have a number of options here in the Object inspector including, minimum and maximum scale, density, fill rate and radius. The scale and density are useful in creating variation between vegetation of the same type, the large the difference between the maximum and minimum scale the greater the size variation. Density is mainly useful for grass and shrubs, allowing you to have dense or sparse grass/shrubs. Have a play with these settings in the lab and see what happens. Once placed a tree/grass can be moved or deleted by selecting them and moving them with the move tools or pressing delete. You will probably also find that you grass/trees don't sit directly on the ground, you will have to move them so you don't end up with floating grass.

If you can't see them when you first paint them on check your render view, it probably set back to 20m (Very small) and the vegetation is just not rendering.

Lighting

Now as we have sunlight we don't have to add another light source to external areas, though we can if we want. However if you have water in your level I recommend you use a light probe, it generates a reflection of the sky and surrounding objects in the water. To do this select the Water Mesh in the Resource Tree, Right click and Select **Insert > New Light Probe**. You might have to look around a bit to find it, it looks like a black spiky ball, you will want to position this in the centre of your body of water just above its surface. Once this is done render both the light maps and the light probe. Once this is done turn off the flat lighting and on the light maps. You may need to tweak and reposition the light probe till you have it how you like.

Note: Trees don't cast shadows but they do reflect in bodies of water when you are using the light probe.

Note: You will need to re-render the light maps after any changes you make to the area.

Models

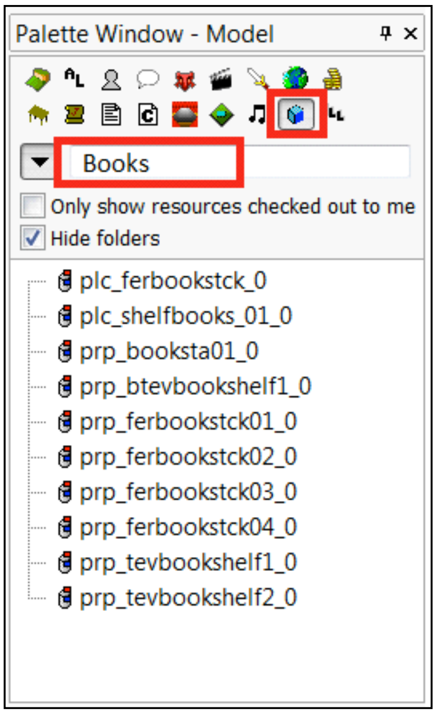
As the Levels are static you can only place on them models NOT placeables. Click on the Model placement button then go to the palette window and click on the models button (blue cube). By default there are no folders in this area, however you can set up fake folders by clicking **Tools > Options > Palette Window > Fake Folders = True**. Now you can filter and search for any models you want and place them in your area. There are a huge range of models you can see them all here: http://social.bioware.com/wiki/datoolset/index.php/Model_list, once you have placed them re-render your light maps and have a look.

Correction to Last week: The Interior tiles are scattered within the list, you just have to find them, searching for floor/wall etc.

This Lab, start building levels, both interiors and exteriors, we will get to exporting levels, walk mapping and areas next week.

Tutorial 3
Path finding, Exporting Levels, Areas, Props & Items

By now you should be able to create you own basic levels. Today I am starting you with a level that is almost ready to go into the area editor. Place the file InteriorLevel.lvl in your toolset directory [Your User]/Bioware/Dragon Age/addins/[Your Module Name]/assets/art. You do not have to place it here but it is useful for convenience sake. Once you have done that start up the toolset, open [Your Module] then open the InteriorLevel.lvl. In this case we are dealing with an interior as they are less prone to error and are simpler to export. When dealing with exteriors it can be a bit trickier.


Palette Window - Model

Books

Only show resources checked out to me

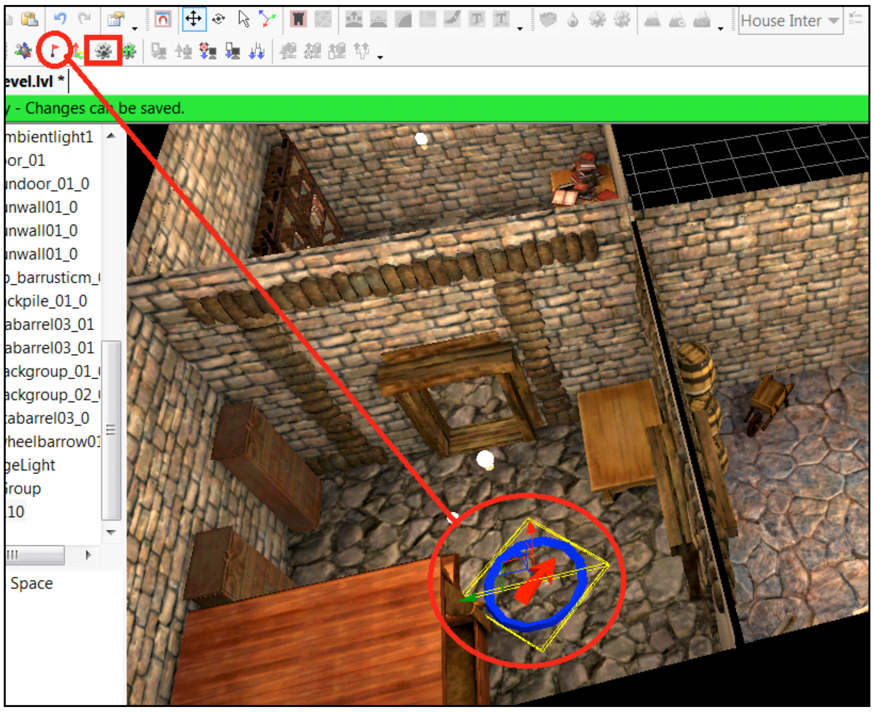
Hide folders

- plc_ferbookstck_0
- plc_shelfbooks_01_0
- prp_booksta01_0
- prp_btevbookshelf1_0
- prp_ferbookstck01_0
- prp_ferbookstck02_0
- prp_ferbookstck03_0
- prp_ferbookstck04_0
- prp_tevbookshelf1_0
- prp_tevbookshelf2_0

First thing you will notice about this level is that it is populated with basic models. Lets add some more go into the models menu. Here you can find the static objects that you will decorate your level with. To find something you want search through the models menu. I am going to look for some more books.

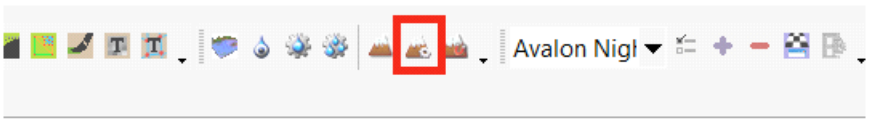
Click on the object and move your mouse into the level view the object should be visible, now left click to place. The trick when you are looking for resources in the DA Toolset is thinking of as many words as possible for what you are looking for, or for things that will look like what you are looking for e.g. if you are looking for a bench for a kitchen, look for a bar or alter or table and find the object that best suits what you are looking for.

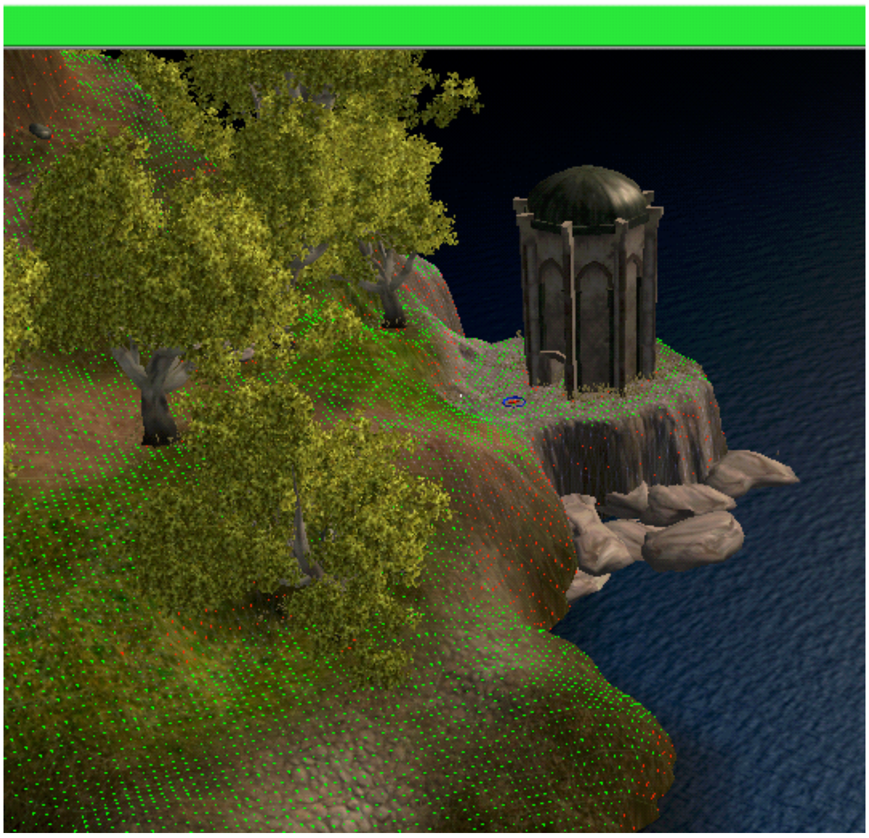
NOTE: These models are static (you cannot interact with them) if you want to put an object in the level you interact with you need to wait till it is imported into an area.



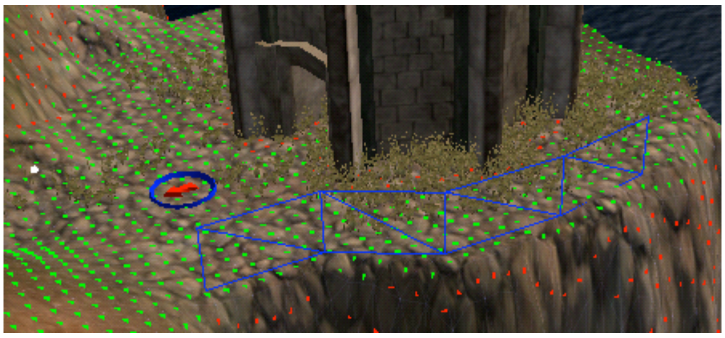
Next thing we need to do is sort out the walk mesh. A walk mesh defines where a player can and can't walk. The first thing we need to do is define a starting point. This is gives the walk mesh rendering tool a point from which to start the mapping.

Now click the "Generate pathfinding for active area" button, it looks like a grey cog with a footprint in it. It will take a while but once it is done your level will be covered in red and green dots. They define the walk able and unwalkable vertexes. For interiors this is simple, however when you do this for exteriors it can be allot messier. If you wanted to tweak the walk mesh in an exterior you use "Build Terrain Collision Tool":

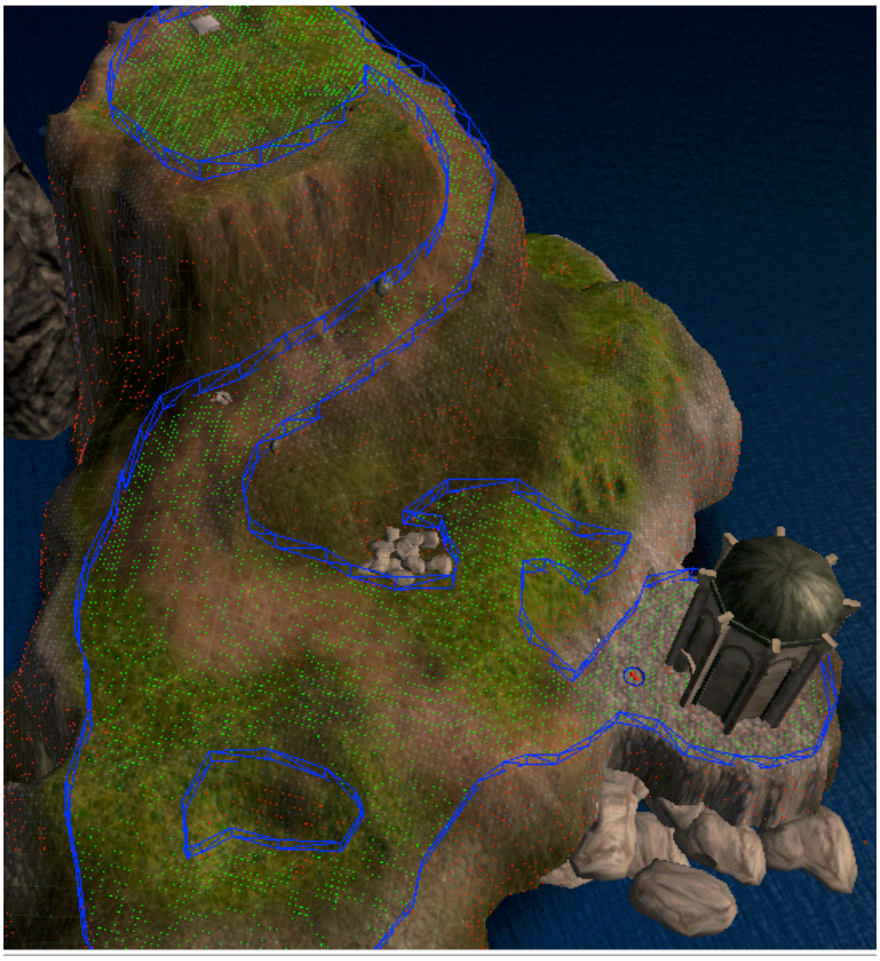




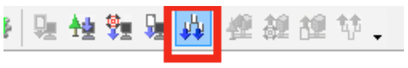
This will allow you to 'draw' a border around the places where you want the player to walk.



In this tool you use left click to 'draw' one of these little walls and right click to remove it. Once you have finished your wall you can click the "Generate pathfinding for active area" button again and it will map out the new walk mesh.

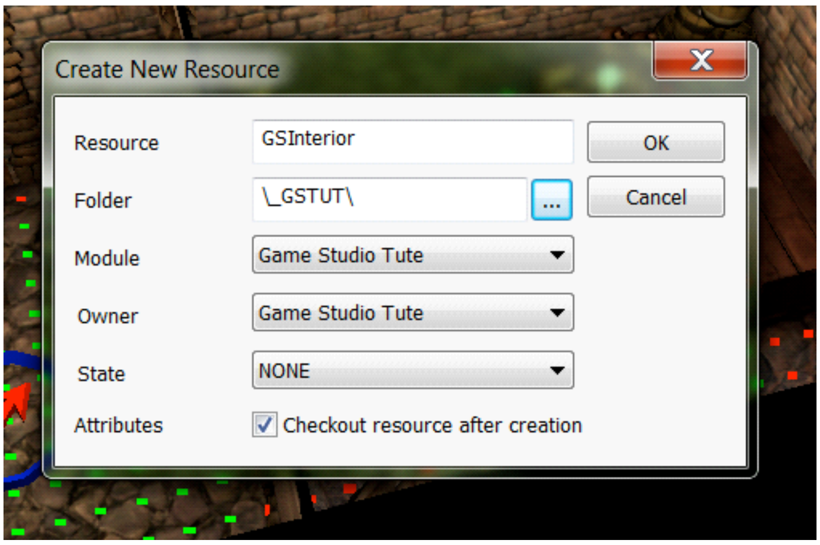


Now you have your lights, objects and path finding done you can now export your level. I recommend re-rendering your light maps, light probes and path finding if you hadn't done it recently just so you know it how you want it. Then export the level!



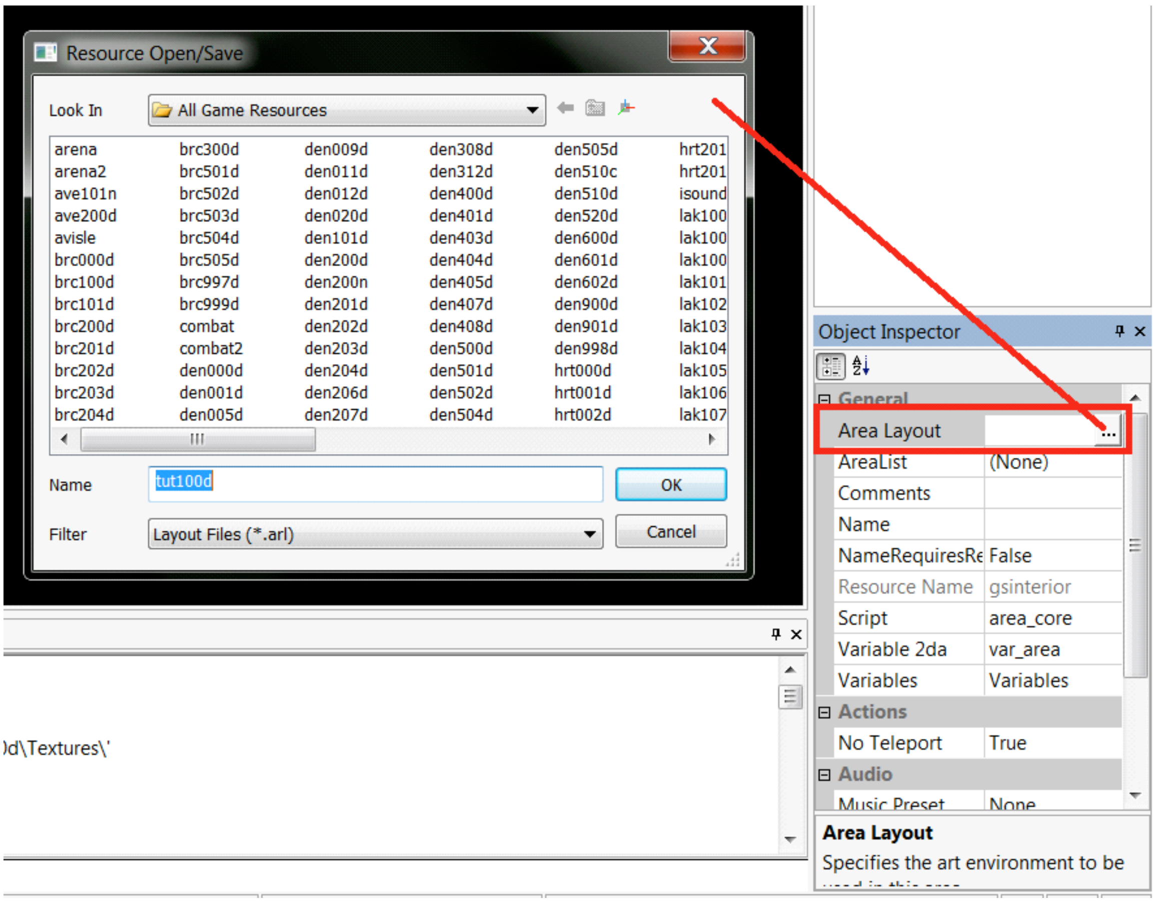
Click on the "Export all to Local Post" this will take a while and the bigger the level the longer it will take. It is also here that problems arise so but be patient. If you come to issues sometimes it takes a toolset reinstall or remapping. It can be a pain so just stick with it, it is worth the battle. When going between computers you may have some issues transferring the area layouts. Make sure you keep the level files with you mod files so if there are any issues you can simply re-export the level.

Now you have exported your level it is time to place it in an area. To do that create a new area File > New > Area. Make sure you put the area into your own mod directory; it makes it easier to keep tack of your resources.


Create New Resource

Resource	GSInterior	OK
Folder	_GSTUT\ ...	Cancel
Module	Game Studio Tute	
Owner	Game Studio Tute	
State	NONE	
Attributes	☑ Checkout resource after creation	

It is important that you remember the Layout Name you chose when you set up your area; we will need this when we choose the area layout. Go to your Area's properties in the Object inspector and click on the Area Layout Box.


Resource Open/Save

Look In: All Game Resources

```
arena     brc300d   den009d   den308d   den505d   hrt201
arena2    brc501d   den011d   den312d   den510c   hrt201
ave101n   brc502d   den012d   den400d   den510d   isound
ave200d   brc503d   den020d   den401d   den520d   lak100
avisle    brc504d   den101d   den403d   den600d   lak100
brc000d   brc505d   den200d   den404d   den601d   lak100
brc100d   brc997d   den200n   den405d   den602d   lak101
brc101d   brc999d   den201d   den407d   den900d   lak102
brc200d   combat    den202d   den408d   den901d   lak103
brc201d   combat2   den203d   den500d   den998d   lak104
brc202d   den000d   den204d   den501d   hrt000d   lak105
brc203d   den001d   den206d   den502d   hrt001d   lak106
brc204d   den005d   den207d   den504d   hrt002d   lak107
```

Name: tut100d — OK

Filter: Layout Files (*.arl) — Cancel

)d\Textures\'

Object Inspector

⊟ General	
Area Layout	...
AreaList	(None)
Comments	
Name	
NameRequiresRe	False
Resource Name	gsinterior
Script	area_core
Variable 2da	var_area
Variables	Variables
⊟ Actions	
No Teleport	True
⊟ Audio	
Music Preset	None

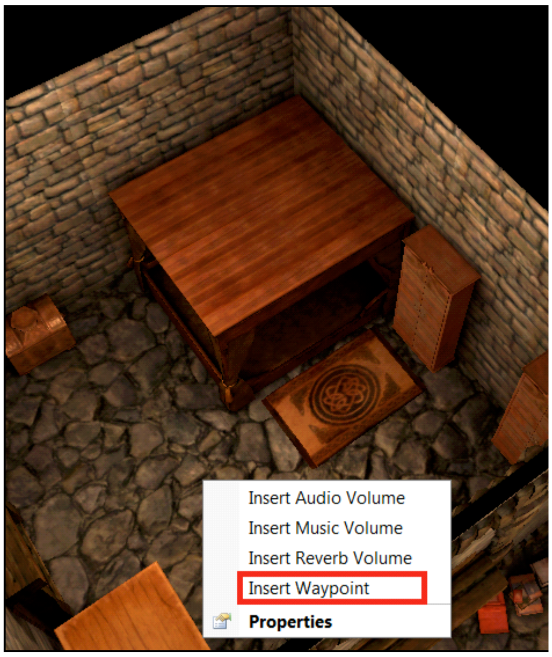
Area Layout
Specifies the art environment to be used in this area

Select your layout; it will load; now you can start the next stage of your mods development.

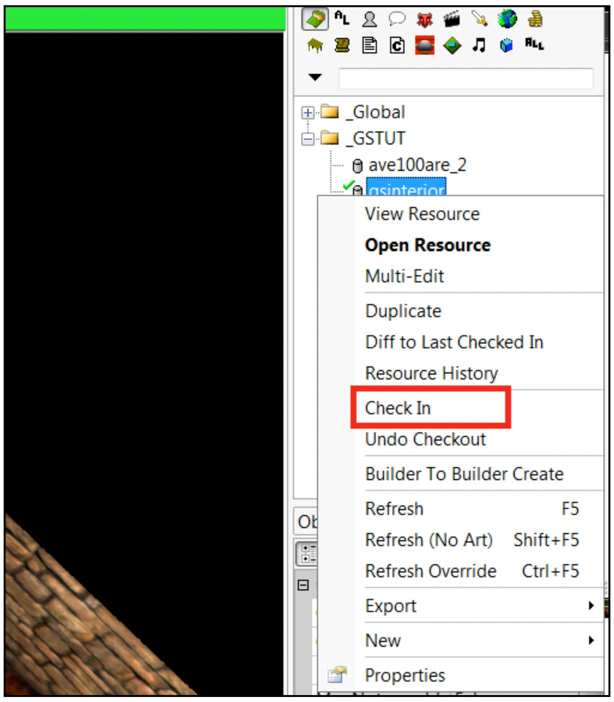
Areas

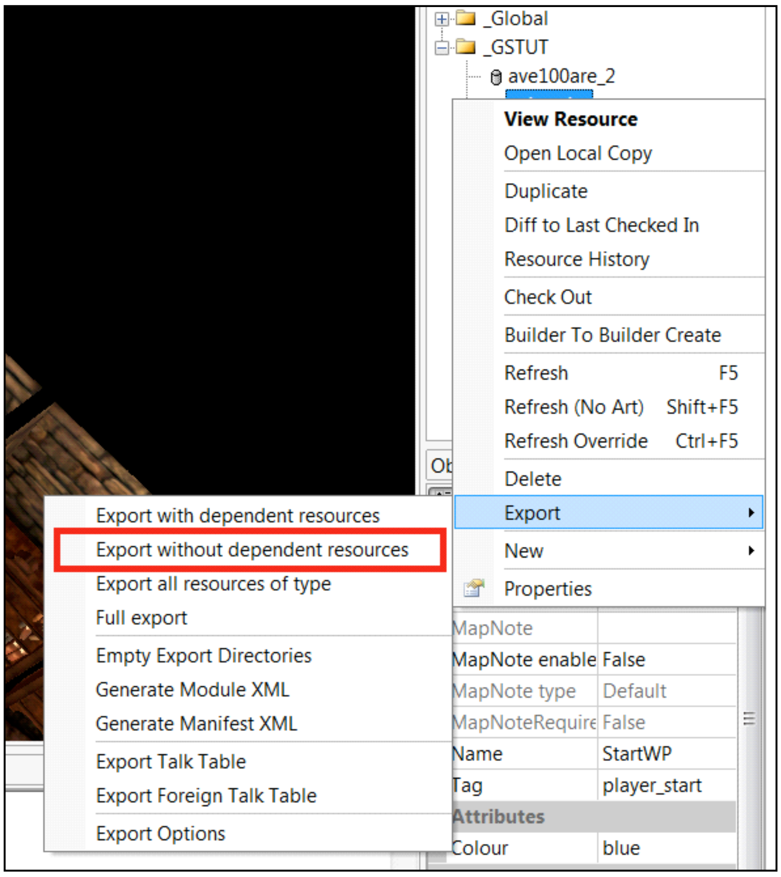
Playing Your Area

We covered this in the scripting tutorials but I will go through it again.



1. Right click In the area and select Insert Waypoint
2. Now you can place the waypoint wherever you want the player to start, it must be within a walk able area.
3. Go to the Tag of the waypoint and call it "player_start", Go to the Waypoint Name and call it "StartWP"
4. Go to File > Manage Modules. Select [Your Module] and click on the properties button
5. Set the Starting Area to the area you have chosen in my case it will be "gsinterior" and set the starting waypoint to "player_start" then OK.
6. Check In the area by right clicking on the resource in the resource tree and click "Check In"
7. Once the Resource is checked in, you can export it. Right Click on the resource > Export > "Export without dependent resources"
8. Now If you are running this at home all you need to do is launch Dragon Age Origins and click on "Other Campaigns" and there you have it your first playable level.



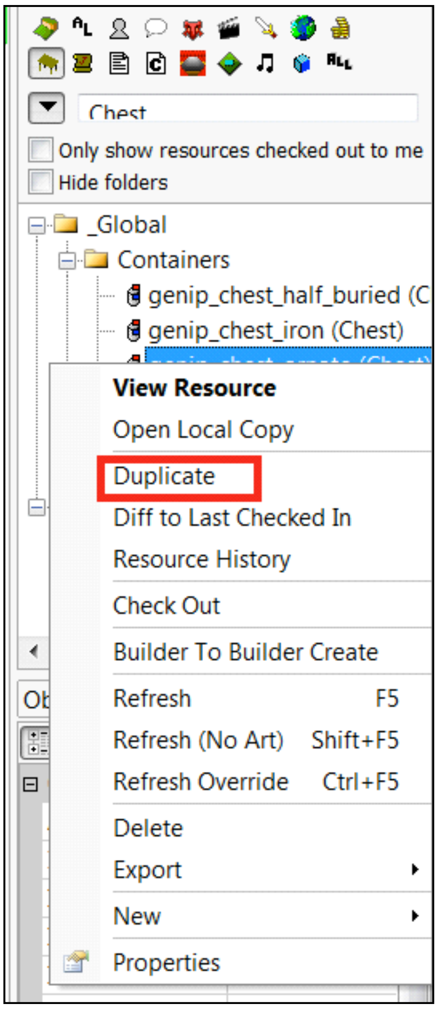


MapNote	
MapNote enable	False
MapNote type	Default
MapNoteRequire	False
Name	StartWP
Tag	player_start
Attributes	
Colour	blue

Placeables

Areas are made up of interactive objects, such as doors chests etc. To start with we are going to create a door placeable. It is important you get used to the process we use here both for the first and second variations of placeable item as this process is the same one you will use for items, characters etc.

1. File > New > Placeable
2. As before make sure to place it in your module folder, give it the name of InteriorDoor1 or some name relevant to your module. It would probably also be a good idea to place the door within a door or transitions folder, depending on what sort of door you are creating.
3. By default the start up placeable is a door. There are two types of doors in the DA toolset ones that simply open and others that work as area transitions. We are looking to have doors that just open but if you were looking to make an area transition. Change its appearance to one that starts with the prefix Area Transition.
4. For the moment we will simply save the place able, check it in (Right click > Check In) and place it within the area. Select the resource in the resource tree and place it in the area. Rotate and position it to suit, remember that on simple doors the arrow points in the direction the door will open for transition al doors it points in the direction you want the player to go.



Now lets try making a lootable object, in this case we will make a chest. Now we could go File > New >Placeable again, however there is an easier way. Go to the placeables menu and do a search for a chest. Find a chest you like then right click > Duplicate.

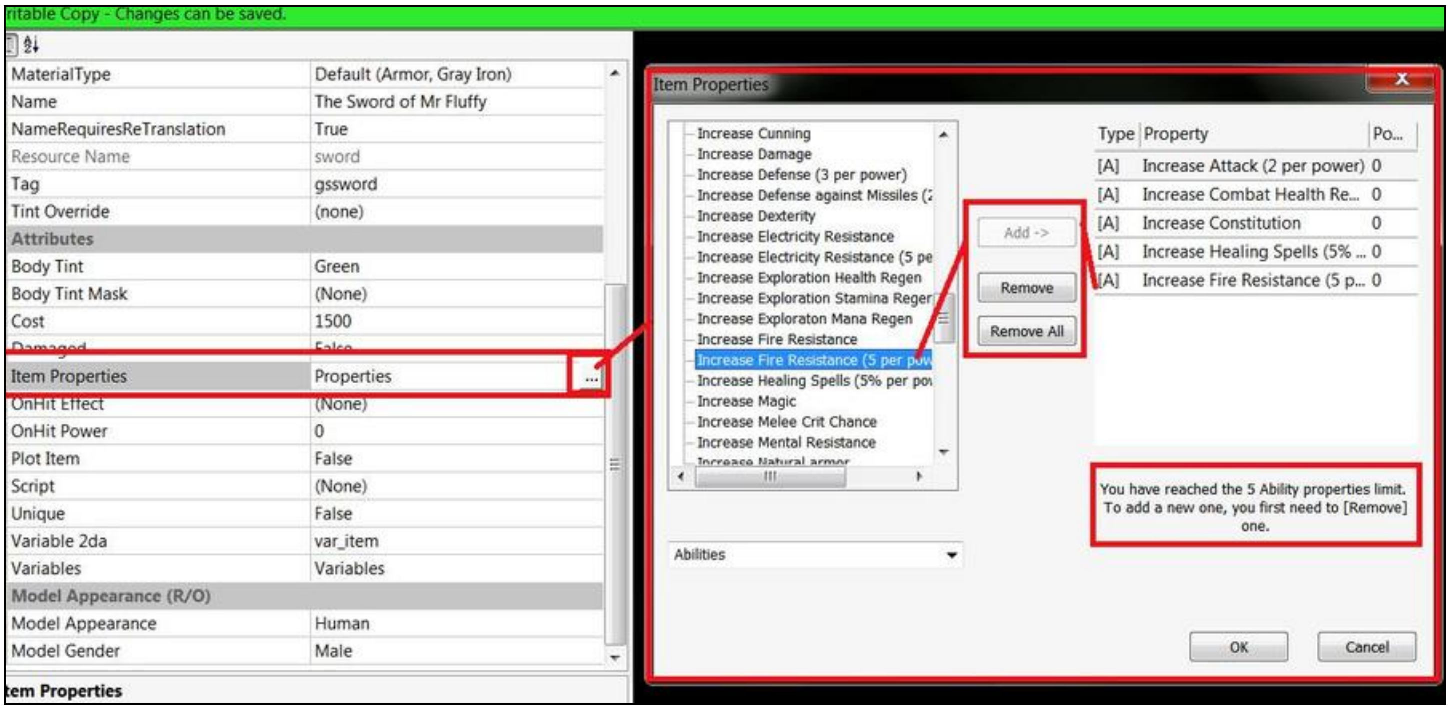
This will create a new placeable chest with all the associated behaviors and scripts of a chest, remember to save it in your mod's file directory. This will open up the placeables editor. Go down to Inventory List and click the "..." button that comes up. Now you can add items to the inventory, I am going to add a dagger. I am going to go into _Global > Weapons, melee > Daggers and add a dagger. Once I have added the item to the inventory, saved the place able and checked it back in I can place it into the area. Now the dagger is in the chest and you can loot it. Run the game and check it out.

This process can be used for all placeables, duplicate add them to your mod. If you have decided to edit the object after you have checked it in merely check it out (Right Click > Check out) and work on it, but remember to save it and check it back in so you don't loose any of your work. If you are planning to manipulate any of the objects with scripts it is important to give them a unique tag and to associate the script with them in the placeable editor.

Items

We have placed items in a container, but what if we needed to make our own items, such as tinted armor, a key or a special weapon. This is how you would go about it:

1. File> New >Item, in this case we are going to work with the default item, a long sword. Remember to place the item in your mods folder.
2. Set the tag to something relevant to your mod. You can also set the description and item name here e.g. "The Sword of Mr Fluffy"
3. Now you have a new long sword. First thing I want to do is alter its appearance you can do this using the "Item Variation" property, pick one you like, I am going with Long sword 4.
4. Set the base cost for the item, it is a good idea to base it on similar items, in this case we will use the long sword cost of 1500cp
5. Item Properties is where the fun really begins. Here you can add special attributes to the item. Have a look through the list.
6. Save and Check In your new Item.
7. Now you should be able to add this to your chest's inventory. I found that I had to reboot my toolset for the inventory of the chest to see my new items, but hopefully you don't have to. Add **the item to the inventory of the chest and test it out.**


ritable Copy - Changes can be saved.

MaterialType	Default (Armor, Gray Iron)
Name	The Sword of Mr Fluffy
NameRequiresReTranslation	True
Resource Name	sword
Tag	gssword
Tint Override	(none)
Attributes	
Body Tint	Green
Body Tint Mask	(None)
Cost	1500
Damaged	False
Item Properties	Properties
OnHit Effect	(None)
OnHit Power	0
Plot Item	False
Script	(None)
Unique	False
Variable 2da	var_item
Variables	Variables
Model Appearance (R/O)	
Model Appearance	Human
Model Gender	Male

tem Properties

Item Properties

- Increase Cunning
- Increase Damage
- Increase Defense (3 per power)
- Increase Defense against Missiles (2
- Increase Dexterity
- Increase Electricity Resistance
- Increase Electricity Resistance (5 pe
- Increase Exploration Health Regen
- Increase Exploration Stamina Reger
- Increase Exploraton Mana Regen
- Increase Fire Resistance
- Increase Fire Resistance (5 per pow
- Increase Healing Spells (5% per poi
- Increase Magic
- Increase Melee Crit Chance
- Increase Mental Resistance
- Increase Natural armor

Type	Property	Po...
[A]	Increase Attack (2 per power)	0
[A]	Increase Combat Health Re...	0
[A]	Increase Constitution	0
[A]	Increase Healing Spells (5% ...	0
[A]	Increase Fire Resistance (5 p...	0

You have reached the 5 Ability properties limit. To add a new one, you first need to [Remove] one.

Abilities

USEFUL LINK: Re-tinting Items

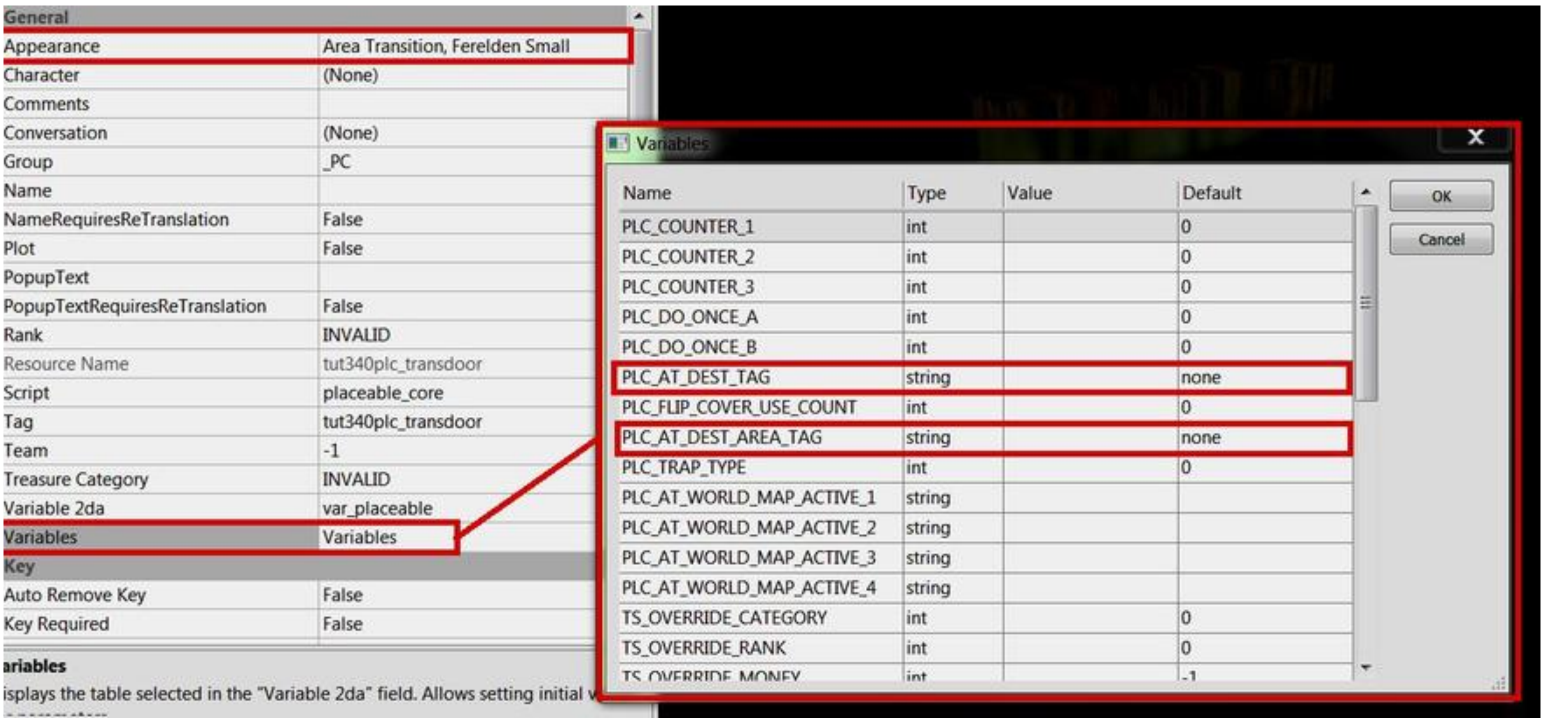
http://social.bioware.com/wiki/datoolset/index.php/Tutorial:_Creating_recolors_of_existing_items

Tutorial 04 - Area Transitions, Conversations Part1

Area Transitions

Last week I mentioned setting up area transition doors , through creating a new placeable and then using the Appearance = "Area Transition, [Door Name]". In this weeks example you can see this in action. When you are ready to set up the transition there are two things you need to do. If yopu want the transition to be two way, e.g. you can go out and come back in, you need to set up a Waypoint on the inside of your area near the door (or where ever you want them to transition to) by **right clicking > insert Waypoint**. You then need to change two of the variables in the doo's variables list by clicking on the doors variables property.

- PLC_AT_DEST_AREA_TAG - tag of the destination area
- PLC_AT_DEST_TAG - tag of the destination waypoint within the destination area



General	
Appearance	Area Transition, Ferelden Small
Character	(None)
Comments	
Conversation	(None)
Group	_PC
Name	
NameRequiresReTranslation	False
Plot	False
PopupText	
PopupTextRequiresReTranslation	False
Rank	INVALID
Resource Name	tut340plc_transdoor
Script	placeable_core
Tag	tut340plc_transdoor
Team	-1
Treasure Category	INVALID
Variable 2da	var_placeable
Variables	Variables
Key	
Auto Remove Key	False
Key Required	False

ariables
isplays the table selected in the "Variable 2da" field. Allows setting initial v

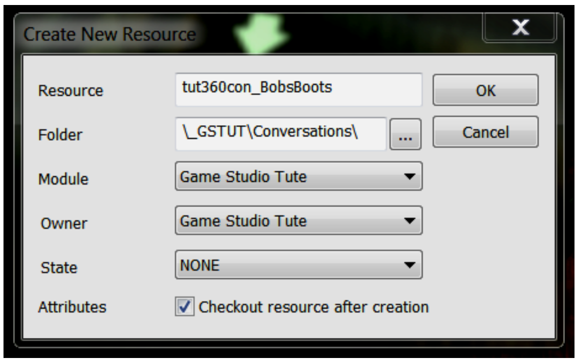
Name	Type	Value	Default
PLC_COUNTER_1	int		0
PLC_COUNTER_2	int		0
PLC_COUNTER_3	int		0
PLC_DO_ONCE_A	int		0
PLC_DO_ONCE_B	int		0
PLC_AT_DEST_TAG	string		none
PLC_FLIP_COVER_USE_COUNT	int		0
PLC_AT_DEST_AREA_TAG	string		none
PLC_TRAP_TYPE	int		0
PLC_AT_WORLD_MAP_ACTIVE_1	string		
PLC_AT_WORLD_MAP_ACTIVE_2	string		
PLC_AT_WORLD_MAP_ACTIVE_3	string		
PLC_AT_WORLD_MAP_ACTIVE_4	string		
TS_OVERRIDE_CATEGORY	int		0
TS_OVERRIDE_RANK	int		0
TS_OVERRIDE_MONEY	int		-1

Once this is done you can check in the resources and place the doors and your transitions should be smooth. Most placeables can serve as area transitions, the reason we use the area transition door is so it doesn't open into the infinite blackness of the end of game world. Theoretically you could easily hook an area transition to a bookshelf or any other placeable object using the two variables listed above.

Conversations

This week we are skipping ahead to conversations as most teams were interested in using them as a pivotal part of their game play. Conversations are an important part of the RPG style game and subsequently the DAO engine has a comprehensive conversation editor. However it doesn't function totally on its own there are number of other tools that link in with the conversations editor that make the conversations side of DAO work. They include Creatures, Plots, Scripts and Stages but first let's get into basic conversations.

DA Conversations 101



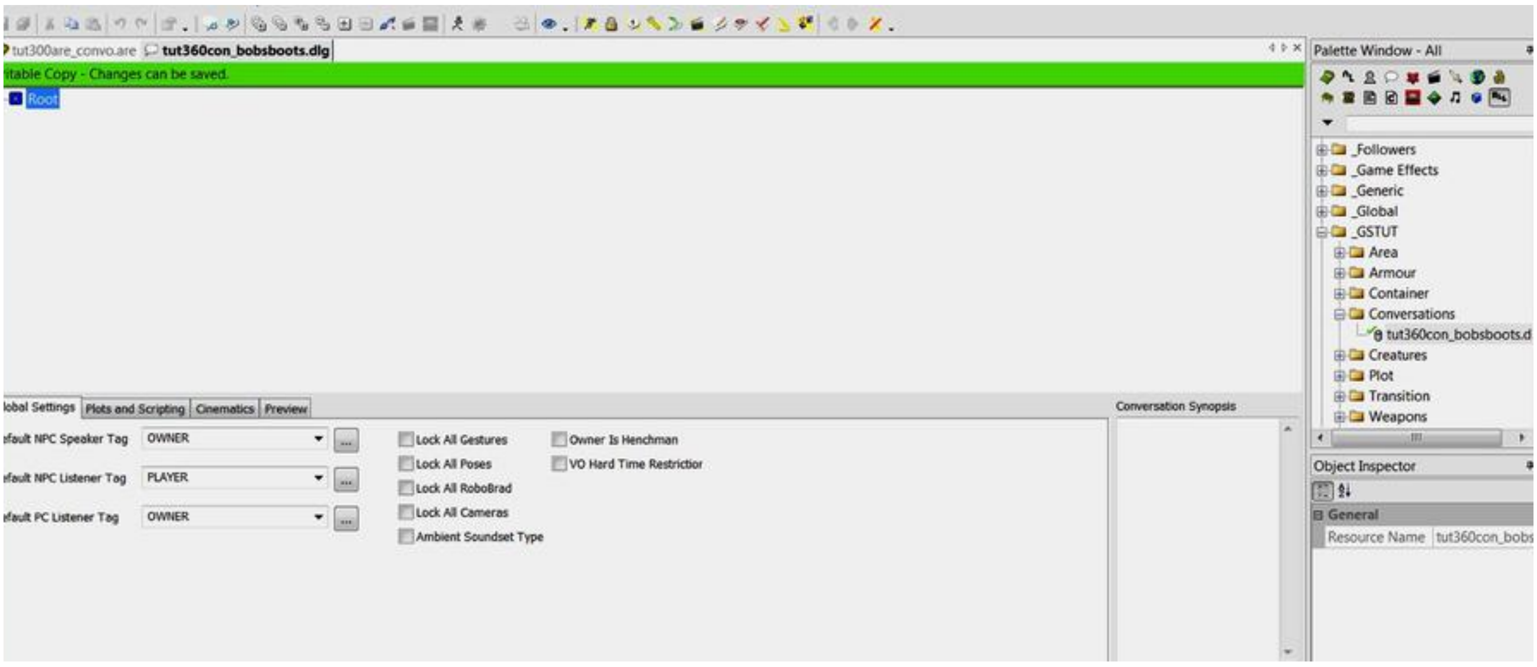
Create New Resource

Resource	tut360con_BobsBoots
Folder	_GSTUT\Conversations\
Module	Game Studio Tute
Owner	Game Studio Tute
State	NONE
Attributes	[✓] Checkout resource after creation

Before you even think about building a conversation you need to think about the way conversations work in RPGs. Many of you will have started working on linear conversations and while this allows you to get from point a to point , it doesn't allow allot of room for the player to customise their game play experience. Conversations in the toolset work on a tree method it starts at the root and branches out, it can loop back on itself and/or can go off on varying tangents. A good way to start thinking about writing your conversation in trees is start with 2-3 different PC types such as: the always good hero, the angry hates the world anti-hero and/or comic relief hero. Start writing the conversation with this in mind write a response suited to each PC type and the NPC responses. This allows you to have a linear style conversation while still giving the player a unique game play experience.

To create a new conversation go to **File > New > Conversation**, create the resource, remembering to put it in your mod's directory.

Now you are in the conversations editor. I do not have time to go through all of the menu items in this tute however on the next page is an image from the wiki that shows the menu items and their functions. Now to start off with just want you to click on the root tag (by default it is selected when you create the conversation) and have a look at the screen. Below the main conversation window you have a tabbed pane, this is where the extra functionality is set.



tut300are_convo.are | tut360con_bobsboots.dlg

itable Copy - Changes can be saved.

Root

lobal Settings | Plots and Scripting | Cinematics | Preview

efault NPC Speaker Tag	OWNER
efault NPC Listener Tag	PLAYER
efault PC Listener Tag	OWNER

[ ] Lock All Gestures
[ ] Lock All Poses
[ ] Lock All RoboBrad
[ ] Lock All Cameras
[ ] Ambient Soundset Type
[ ] Owner Is Henchman
[ ] VO Hard Time Restrictior

Conversation Synopsis

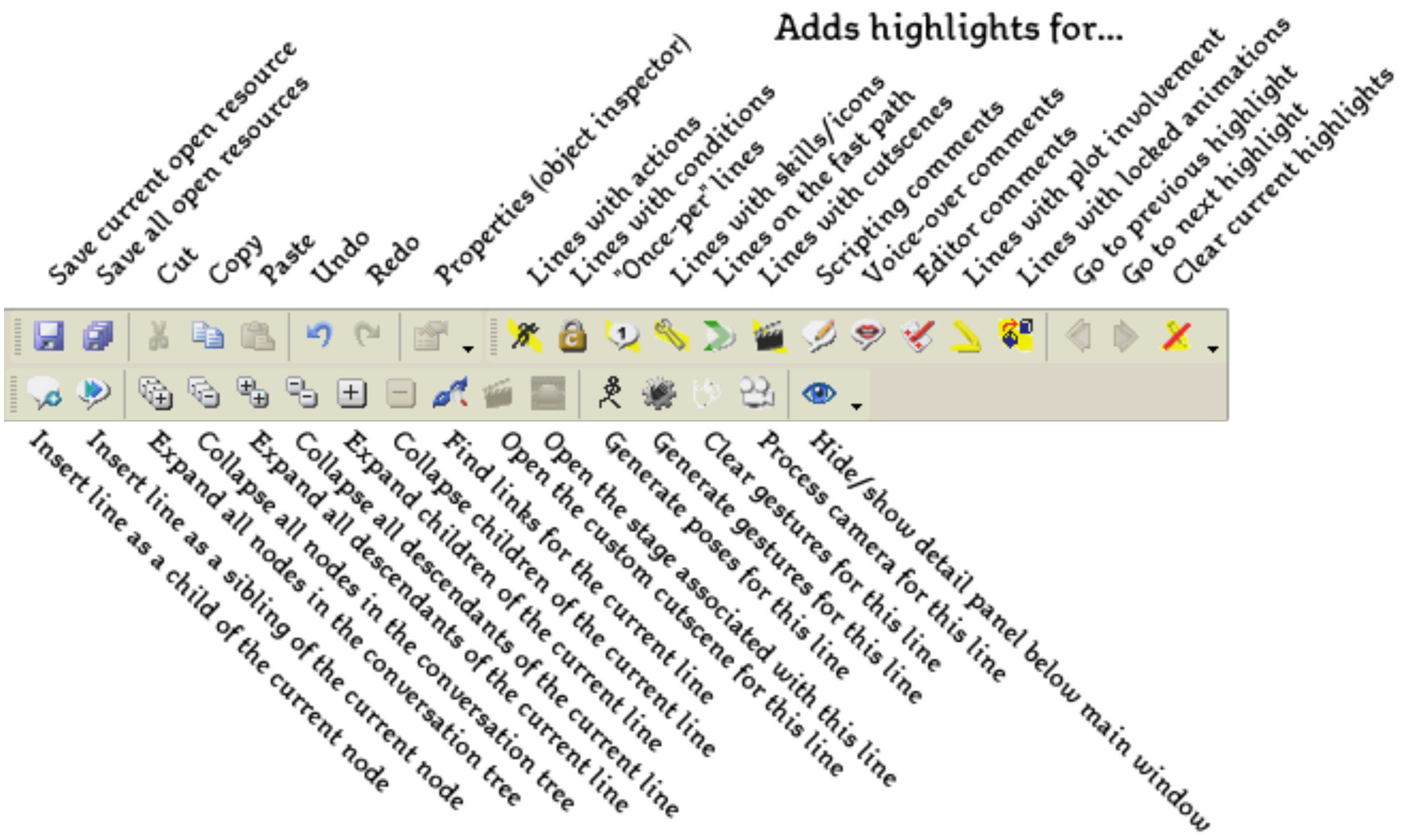
Palette Window - All

- _Followers
- _Game Effects
- _Generic
- _Global
- _GSTUT
  - Area
  - Armour
  - Container
  - Conversations
    - tut360con_bobsboots.d
  - Creatures
  - Plot
  - Transition
  - Weapons

Object Inspector

General

Resource Name	tut360con_bobs


Adds highlights for...
Save current open resource
Save all open resources
Cut
Copy
Paste
Undo
Redo
Properties (object inspector)
Lines with actions
Lines with conditions
"Once-per" lines
Lines with skills/icons
Lines on the fast path
Lines with cutscenes
Scripting comments
Voice-over comments
Editor comments
Lines with plot involvement
Lines with locked animations
Go to previous highlight
Go to next highlight
Clear current highlights
Insert line as a child of the current node
Insert line as a sibling of the current node
Expand all nodes in the conversation tree
Collapse all nodes in the conversation tree
Expand all descendants of the current line
Collapse all descendants of the current line
Expand children of the current line
Collapse children of the current line
Find links for the current line
Open the custom cutscene for this line
Open the stage associated with this line
Generate poses for this line
Generate gestures for this line
Clear gestures for this line
Process camera for this line
Hide/show detail panel below main window

Global Settings

This is where you set the main speaker and listener for the conversation, you can add in extra characters later, but these are the main ones. By default the main speaker is the NPC the conversation is associated with and the default listener is the PC. There are a bunch of other options in this window such as locking some of the features like Cameras or setting the OWNER to one of the player's Henchmen, I won't be going into these today if you are interested in exploring them try them out and see what they do.

Plots and Scripting

This is where you can set default plot or script events at the end of the conversation, this means that no matter where or how the conversation ends certain things will take place, you don't need to apply them to all the different end nodes of your conversation.

Cinematics

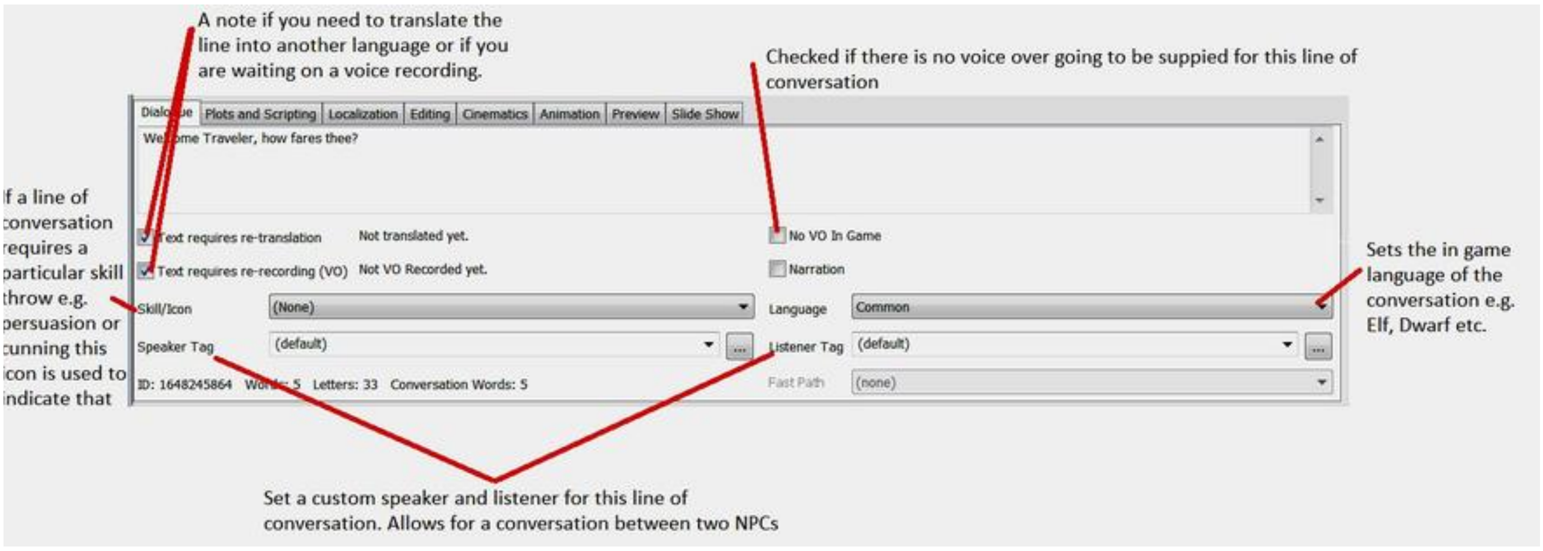
Where you associate your stage with this conversation.

Preview

I haven't used this aspect of the toolset but it appears to be a quick way to link up a conversation to an area and NPC to preview the conversation.

Writing the Conversation

You now have a rough idea about what you are going to write, so right click on 'Root' in the conversation window and "Insert Line". If you de-select it you will see it is red. This is the first NPC line select it again and write a welcome in the dialog box. If you examine the tabbed pane down the bottom it has changed you now have a range of extra features.



A note if you need to translate the line into another language or if you are waiting on a voice recording.

Checked if there is no voice over going to be suppied for this line of conversation

If a line of conversation requires a particular skill throw e.g. persuasion or cunning this icon is used to indicate that

Sets the in game language of the conversation e.g. Elf, Dwarf etc.

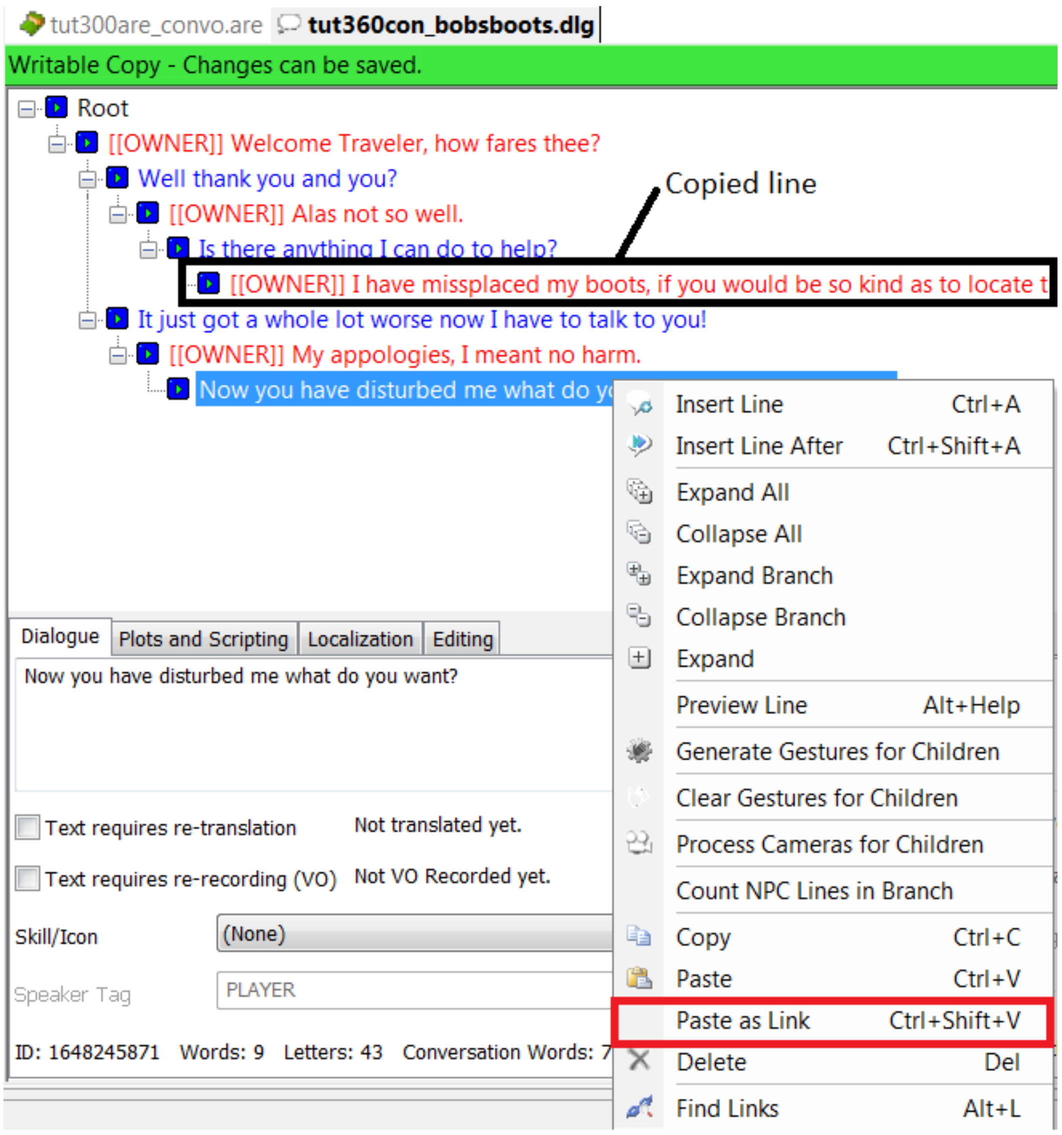
Set a custom speaker and listener for this line of conversation. Allows for a conversation between two NPCs

Welcome Traveler, how fares thee?

[✓] Text requires re-translation Not translated yet.
[✓] Text requires re-recording (VO) Not VO Recorded yet.
[ ] No VO In Game
[ ] Narration
Skill/Icon: (None)
Language: Common
Speaker Tag: (default)
Listener Tag: (default)
Fast Path: (none)
ID: 1648245864 Words: 5 Letters: 33 Conversation Words: 5

For the moment we are going to leave these be and just write some more lines. Right click on the line you just wrote and select "Insert Line". This is your first PC Line, you will notice it has [END DIALOGUE] written in it. It is important to note here that only PC lines can end a conversation, if you want to finish the conversation with the NPC just insert an empty PC line like this one. Now we can add some text here I am going to go for the nice hero first and write some pleasant response. Now if you right click on this line and select "Insert Line After" it will add another PC choice, in this one I will add something a little more hostile. If you click on these lines and click "Insert Line" you can add an NPC response and continue on that way. If you are designing a conversation between two NPCs and want to skip the PC line you still have to put it in but just leave it empty and add another NPC line and it will become a 'continue'.

One thing you will find in RPGs is that you will often want to allow the player to investigate things so bring them back to a range of options or bring the conversation back to a single ending point. To do this without having to re-type all the lines the DAO editor has line linking system to create a link select the line you want to link to and copy it either through **ctrl + c** or **right click > copy**. To link the line select the line you wish to link to the line you just copied and paste the link by **right clicking > Paste as Link** or **ctrl + shift + v**.


tut300are_convo.are tut360con_bobsboots.dlg

Writable Copy - Changes can be saved.

```
Root
  [[OWNER]] Welcome Traveler, how fares thee?
    Well thank you and you?
      [[OWNER]] Alas not so well.
        Is there anything I can do to help?
          [[OWNER]] I have missplaced my boots, if you would be so kind as to locate t
    It just got a whole lot worse now I have to talk to you!
      [[OWNER]] My appologies, I meant no harm.
        Now you have disturbed me what do y
```

Dialogue | Plots and Scripting | Localization | Editing

Now you have disturbed me what do you want?

Text requires re-translation — Not translated yet.

Text requires re-recording (VO) — Not VO Recorded yet.

Skill/Icon: (None)

Speaker Tag: PLAYER

ID: 1648245871 Words: 9 Letters: 43 Conversation Words: 7

Copied line
Insert Line Ctrl+A
Insert Line After Ctrl+Shift+A
Expand All
Collapse All
Expand Branch
Collapse Branch
Expand
Preview Line Alt+Help
Generate Gestures for Children
Clear Gestures for Children
Process Cameras for Children
Count NPC Lines in Branch
Copy Ctrl+C
Paste Ctrl+V
Paste as Link Ctrl+Shift+V
Delete Del
Find Links Alt+L

Once it is pasted it will appear grey, if you wish to edit the line of text you will have to go to the actual line and edit it any changes will be reflected in the link. If you wish to delete a link or a line of text all you need to do is **right click > delete** or **select the line and press the delete key** (fn +delete on a boot camped Mac). Finish off the conversation save it and check it in.

In the demo levels I have provided I have created a custom creature, Bob to add a conversation to a creature you need to edit it's template, I will be doing creatures and characters next week so for the moment just use bob. Edit his template and add the conversation you just created to his conversation property. Now you can export the resources and run the game and talk to Bob. Nothing exciting there but a repeated conversation, very rarely will you want a conversation with no game relevance so now we get on to the more interesting bits.

There a number of formatting featured provided for in conversation text formatting such as <emp> </emp> for emphasis you can see a list of them and other in text features here:

http://social.bioware.com/wiki/datoolset/index.php/Conversation#Formatting_tags

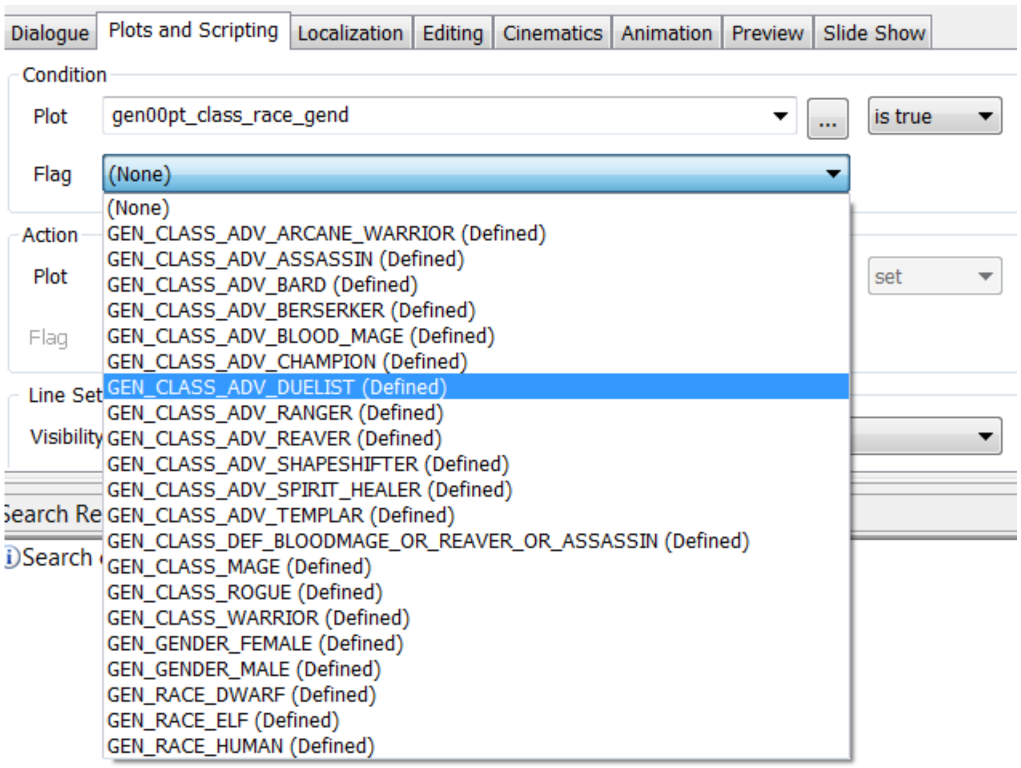
Conditions & Actions

Now in the above section we made a conversation with no special features, most of the special features need to be added through a combination of script and plot driven:

- Conditions - if true/false the line is spoken; and
- Actions - after the line is spoken an action takes place.

Generic Conditions

but there are a range of generic conditions and actions in the DAO editor to get to them click on the ... next to the plot Condition/Action go to _Global > Generic and you will see them. Select one and it will have a range of condition options in the Flag drop down e.g.

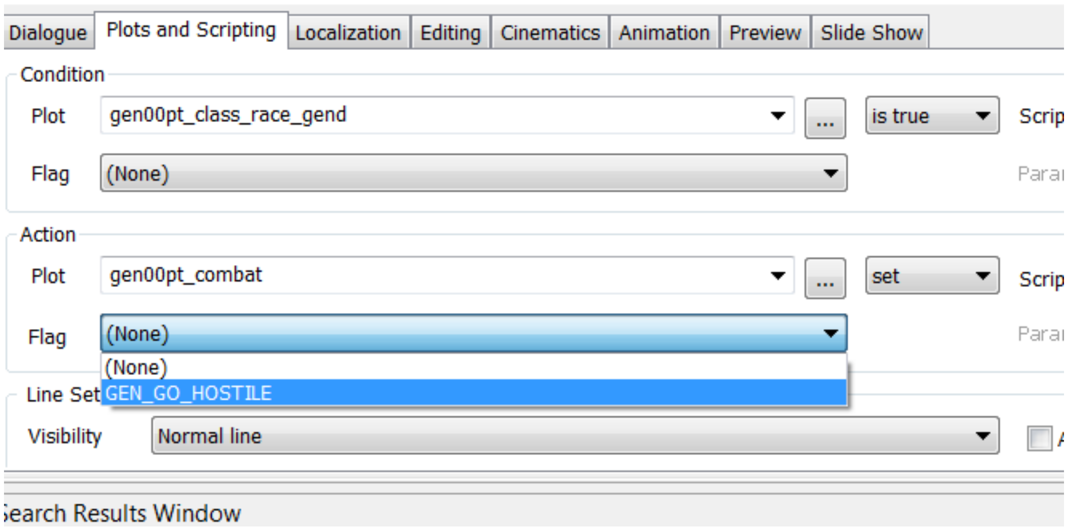

Dialogue | Plots and Scripting | Localization | Editing | Cinematics | Animation | Preview | Slide Show

Condition
Plot: gen00pt_class_race_gend ... is true
Flag: (None)

(None)
GEN_CLASS_ADV_ARCANE_WARRIOR (Defined)
GEN_CLASS_ADV_ASSASSIN (Defined)
GEN_CLASS_ADV_BARD (Defined)
GEN_CLASS_ADV_BERSERKER (Defined)
GEN_CLASS_ADV_BLOOD_MAGE (Defined)
GEN_CLASS_ADV_CHAMPION (Defined)
GEN_CLASS_ADV_DUELIST (Defined)
GEN_CLASS_ADV_RANGER (Defined)
GEN_CLASS_ADV_REAVER (Defined)
GEN_CLASS_ADV_SHAPESHIFTER (Defined)
GEN_CLASS_ADV_SPIRIT_HEALER (Defined)
GEN_CLASS_ADV_TEMPLAR (Defined)
GEN_CLASS_DEF_BLOODMAGE_OR_REAVER_OR_ASSASSIN (Defined)
GEN_CLASS_MAGE (Defined)
GEN_CLASS_ROGUE (Defined)
GEN_CLASS_WARRIOR (Defined)
GEN_GENDER_FEMALE (Defined)
GEN_GENDER_MALE (Defined)
GEN_RACE_DWARF (Defined)
GEN_RACE_ELF (Defined)
GEN_RACE_HUMAN (Defined)

Action
Plot
Flag
set
Line Set
Visibility
Search Re
Search

The same works for generic actions:


Dialogue | Plots and Scripting | Localization | Editing | Cinematics | Animation | Preview | Slide Show

Condition
Plot: gen00pt_class_race_gend ... is true Scrip
Flag: (None) Parai

Action
Plot: gen00pt_combat ... set Scrip
Flag: (None) Parai

(None)
GEN_GO_HOSTILE

Line Set
Visibility: Normal line

Search Results Window

However these generic features are very basic you will definitely want to add more Conditions and Actions using your own Plots (Quests) and Scripts.

For more info on Generic Conditions see:
http://social.bioware.com/wiki/datoolset/index.php/Conversation_plots_and_scripting#Generic_conditions

Plots

Plots are quite a complex topic and have a range of different functionalities, however we will be mainly using them for their quest and journal features to read about the range of plot related features check out:
http://social.bioware.com/wiki/datoolset/index.php/Plot

To create a new Plot go to **File > New > Plot**. Create the new resource remembering to put it into your mods directory I am going to call mine "tut370plt_BobsBoots". **Right Click in the main window and click insert > Main Flag** or **ctrl + m**, you have your first flag. Change the Name from FLAG_0 to FIND_BOOTS with it selected look to the bottom half of the window you will find a box titles Journal Text, type in it the text you want to appear in the journal. Go to the Properties Window and Select the Name field type the name you want to appear in the journal, I will type "Find Bob's Boots." You can also set a number of other properties here, have a look some of them may be useful for you. I will set the Priority for this quest to High, when you are making your mod you can set side quest to a lower priority and main quests to a higher and the journal will sort them in order of priority. Insert 2 more Main Flags

- BOOTS_FOUND, Journal: "You have found Bob's boots, return them to Bob"
- BOOTS_RETURNED, Journal: "You have returned Bob's boots and gained his gratitude"

In the last Flag set the property Final to Yes, this will move the quest from the active quest tree to the completed quests tree. You can also add a reward, the rewards are predefined in the Rewards.xls 2DA file so if you want to create your own you will have to do it through this (see:
http://social.bioware.com/wiki/datoolset/index.php/Rewards.xls & the Plots link above). For the moment I will set the reward on the final entry to the string reward just to show you how it works.

Save it and check it in, now go back to the conversation with Bob. Go to the your final lines and go to the lines Actions > Plot, click on the ... go to the folder you saved the plot we just created in and select it, using the drop box click on FIND_BOOTS.

Now the Conversation gives you the quest, but we want to stop it from trying to give it to you twice. Go to the first line of the tree and add a Condition. Go to Conditions > Plot Select the same plot file we just created and the FIND_BOOTS entry but instead of "is true" select "is false". Now this line will only be said when we don't have that particular journal entry, this is not really the best way to do it as we have more than two entries, however for the moment it will do. Add a new line for Bob from the root "Have you found my boots?" put a yes & no PC reply.

If you save it, check in all the resources, export it and run, the conversation with bob will run he will give you the quest then if you talk to him again he will ask you if you found his boots. Now you could just add the Found boots final quest update to the PC Yes line in the second tree, however you probably want to check if the player actually has the item which means we have to move into the realm of scripts.

Scripts

As the artists and designers here may not really like code, I have created a importable database with a couple useful conversation scripts.

Scripts for conditional statements, which is where you will really want to use them, are a special form of DA function that don't require a main() e.g.:

```
int StartingConditional()
{
    return 1;
}
```

The first script I have given you is "conversation_inventroycheck" it checks the inventory of the PC and returns true if it contains the item of a specified tag, I have also put a link to an example script for checking an entire parties inventory in the comment at the top:

```
int StartingConditional()
{
     object oPC = GetMainControlled();

    /* change the string in the below function to your own
    object's tag*/

     object oItem = GetItemPossessedBy(oPC, "bob_boots");


     if(IsObjectValid(oItem))
        {
            return TRUE;
        }

     return FALSE;
}
```

Import this script from the database and add it to your module folder, go to the conversation tree and select the yes option add the conditional script to it. Add the Final Plot Flag from the Plot we wrote earlier to the actions of this line. Now also add a Plot conditional to Bob's "Have you found them?" line: BOOTS_RETURNED is false. It might also be a good idea to add another line for bob off the root that just says "Thank you for finding my Boots".

Save, check in, export and run.

You will notice that the character is a bit wooden with no animations etc. To fix that you will use stages but as they are a polishing aspect we will be covering them in a couple of weeks.

Tutorial 05

Visual Effects

We have discussed level building and you all should be well on your way to having your levels built. This week I am going to go back to levels a bit and touch briefly on visual effects (VFX) which can be added to you levels to add atmosphere as well as a more dynamic feel. I will also provide a list of links to the VFX parts of the toolset wiki for those interested in using them with characters, spells or creating their own.

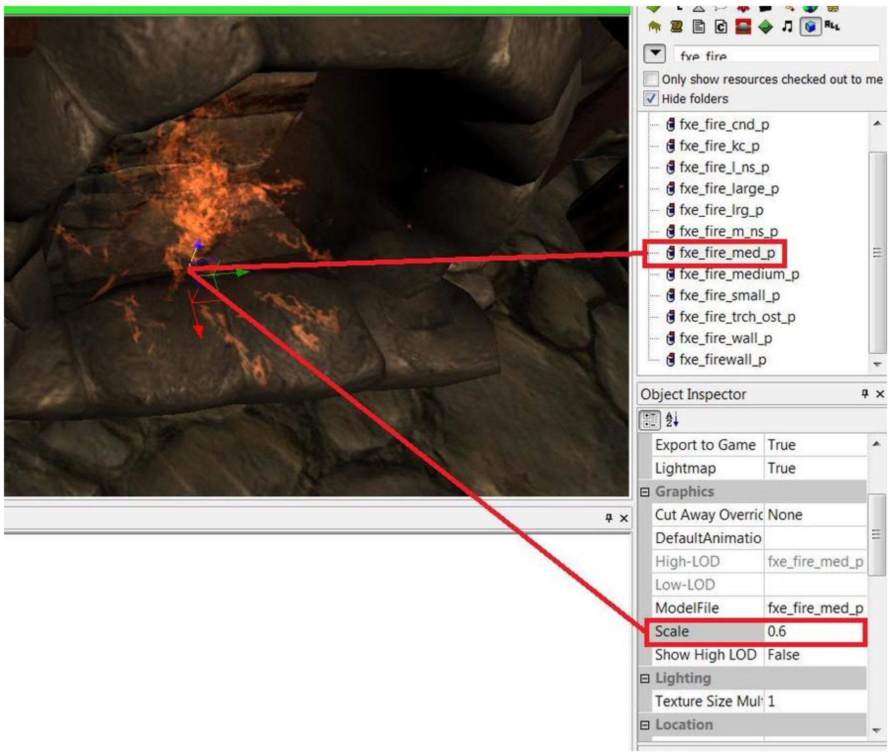
First off I am going to add a fire place to my interior, models = fti_fireplace_a_0. Place it where you want it, we are just using it for a context for the effect. Now go to the models filter and search for fxe_fire. The fxe stands for environmental effects, there are a number of prefixes and suffixes you should be aware of when dealing with effects they are:

Prefixes:

- VFX - general visual effects
- FXA - abilities
- FXE - environmental
- FXC - character
- FXP - placeable
- FXM - animator cut scene vfx
- SR - GUI/selection

Suffixes:

- _p - placeable, can be placed anywhere in the scene.
- _c - crust, should be attached to a model.



fxe_fire

Only show resources checked out to me

☑ Hide folders

- fxe_fire_cnd_p
- fxe_fire_kc_p
- fxe_fire_l_ns_p
- fxe_fire_large_p
- fxe_fire_lrg_p
- fxe_fire_m_ns_p
- fxe_fire_med_p
- fxe_fire_medium_p
- fxe_fire_small_p
- fxe_fire_trch_ost_p
- fxe_fire_wall_p
- fxe_firewall_p

Object Inspector

Export to Game	True
Lightmap	True
Graphics	
Cut Away Overric	None
DefaultAnimatio	
High-LOD	fxe_fire_med_p
Low-LOD	
ModelFile	fxe_fire_med_p
Scale	0.6
Show High LOD	False
Lighting	
Texture Size Mul	1
Location	

We will be dealing with **fxe_ fire_med_p** a medium fire environmental effect placeable . Place your fire in your level and position it within the fireplace. You will notice the fire is too big, however if you go for fxe_fire_small_p the visual effect is too small. So what we will do is scale this effect. once you have placed the effect select it and go to the object Inspector scroll down to scale and set it to 0.6, or whatever best suits your chosen fireplace. I would then add an animated light click on the room that your fire is in right click > new >light. Set the light type to point animated. Set the colour to something with a red-orange tint. I would then go down to the Animated section. Here is where you can set the frequency and intensity of the flicker. I am going to leave most of it as is but change the minimum intensity to 1 so that the difference between the max and min in the flicker is more noticeable. I am also going to set the Light Character property to true so that my character is illuminated in the flickering light. Now you can add your start point and export, when you import it you will have a fireplace that illuminates the room and the characters with a flickering orange light.

VFX Links:

- VFX List: http://social.bioware.com/wiki/datoolset/index.php/VFX_list

- VFX Tutorial: http://social.bioware.com/wiki/datoolset/index.php/VFX_Tutorial
- VFX editor: http://social.bioware.com/wiki/datoolset/index.php/VFX_editor

Creatures & Characters

Creatures and characters are two different things though a Creature may be a Character.

Characters

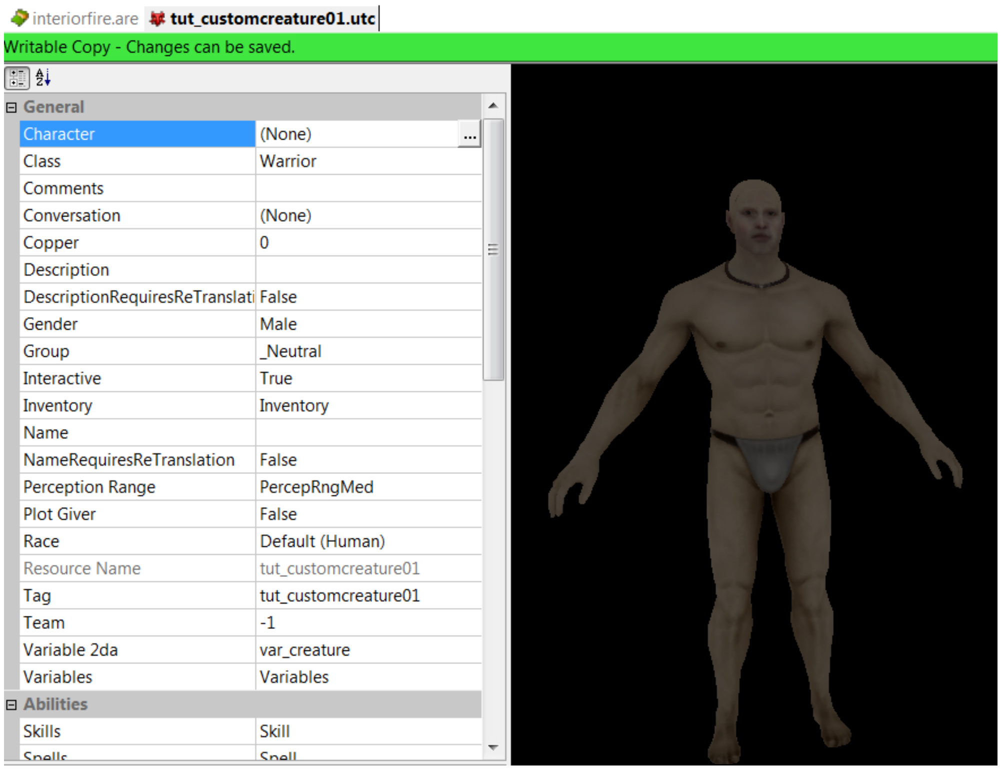
Characters are basically a human focused definition of a creature. It was created for the DA:O development team to define a character's background, family etc for voice actors and other human contributors. It is not used directly by the DA:O toolset. If you are interested in reading more about characters I suggest you look up the character blurb on the wiki: http://social.bioware.com/wiki/datoolset/index.php/Character.

Creatures

http://social.bioware.com/wiki/datoolset/index.php/Creature

Creatures are all the NPCs, animals and variations thereof in the game. If you wish to use a default creature with default behaviour e.g. a hostile darkspawn, then you can simply go to the creatures menu (_Core_creatures > Darkspawn > Hurlocks > Normal > hurlock_axe) and place the creature where you want it, by default they are hostile and will attack as soon as they perceive the player. However if you wanted something a little more customised, you will have to create your own creature, so let's do that.

There are two ways we can do this the first is to find a creature that has the behaviour or appearance we want and then right click and duplicate them or we can go to File > new > Creature. In this case we will create a new creature from scratch.



Character	(None)
Class	Warrior
Comments	
Conversation	(None)
Copper	0
Description	
DescriptionRequiresReTranslati	False
Gender	Male
Group	_Neutral
Interactive	True
Inventory	Inventory
Name	
NameRequiresReTranslation	False
Perception Range	PercepRngMed
Plot Giver	False
Race	Default (Human)
Resource Name	tut_customcreature01
Tag	tut_customcreature01
Team	-1
Variable 2da	var_creature
Variables	Variables
Skills	Skill

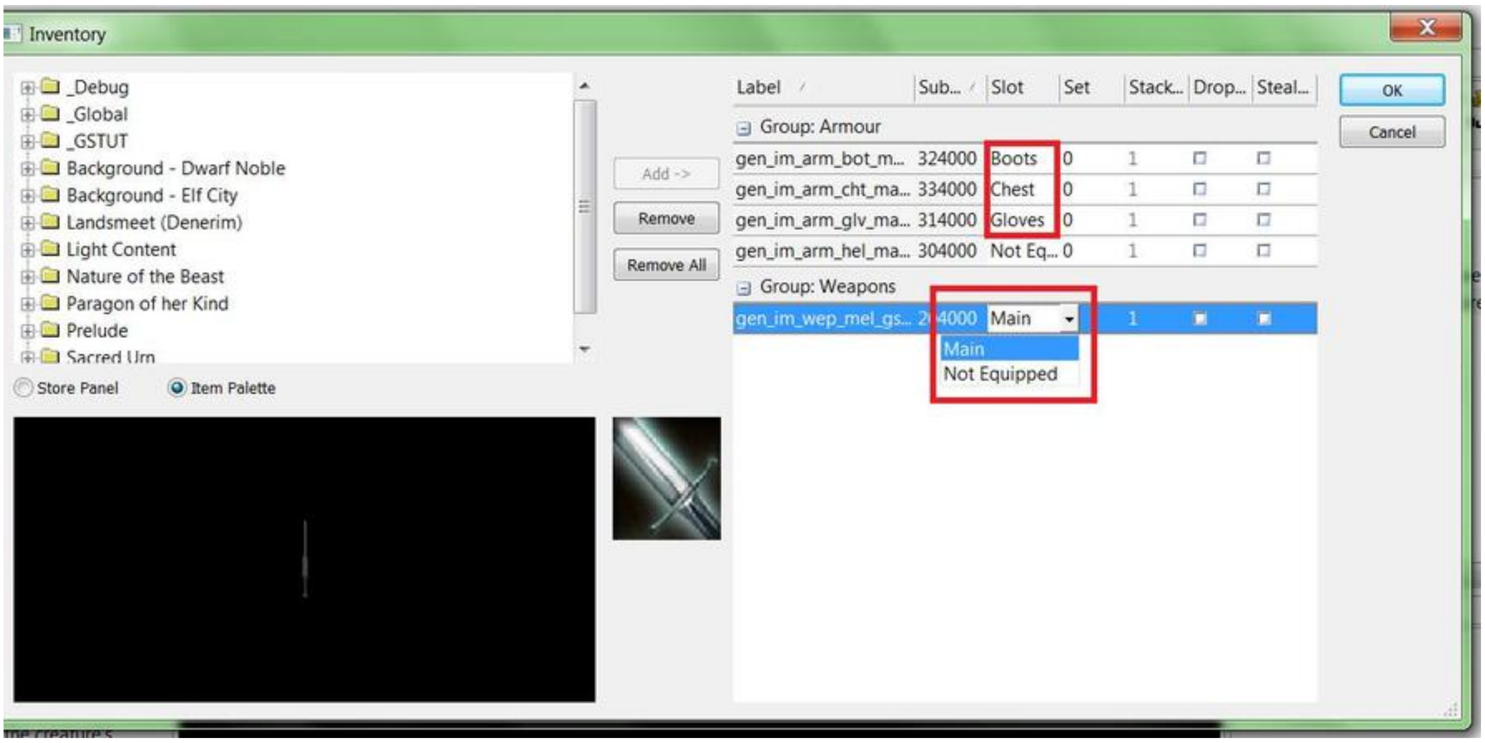
Creating a Creature from Scratch

The first step of creating any creature is working out what you want to create. That is one of the useful parts of the character tool, it allows you to create a full profile for a creature. I would recommend you use it when you are creating your core characters, but it is not necessary. The core thing here is that you need to know what you want to create. I am going to create the first version of a creature that will reoccur throughout my game. This is the profile:

- Name: Mordred
- Gender: Male
- Description: The Son of Arthur and Morgana Le Fey, a vengeful and dark character who will stop at nothing to attain his father's crown.
- Class: Arcane Warrior
- Plot/Quest: Dark victory - Steal Arthur's sacred scabbard. Final boss
- Group/faction: LeFey (new groups created in the 2DA file: http://social.bioware.com/wiki/datoolset/index.php/Creature_group);
- Tag: mordred_openq
- Skills: Master persuasion, master survival, master combat training
- Spells: Aura of might, shimmering shield, fade shroud
- Talents:
    - Assassin: Exploit weakness, mark of death,
    - Dualist: all
    - Warrior: Death blow, perfect strike, precise striking, disengage
    - Two handed: Critical Strike, pommel strike, powerful swings, stunning blows, two handed strength
- Appearance: fair skinned, dark hair and stubble, green eyes, tall, lean
- Min Skill: 8
- Max Skill: 15
- Rank: BOSS
- Plot: True

So now we will implement this.

I generally find working from top to bottom a good idea, others like to set up the appearance then implement the more technical side, it is up to you. There are a few things I want to talk you through the first is equipment. Your creature starts off without any equipment and almost naked, to put clothes on them we need to go to the inventory section. Find the equipment/clothes you would like to equip and place it in the inventory. Once you have placed it in the inventory you can equip to an equipment slot. you can also set it to drop able and or steal able in this case we do not want his gear drop able or steal able as it is the first time we are meeting this character so we don't want our low level PC getting his high level gear. We'll not put on his helmet in this case as we want to have a conversation with him so we will leave that off for the moment.

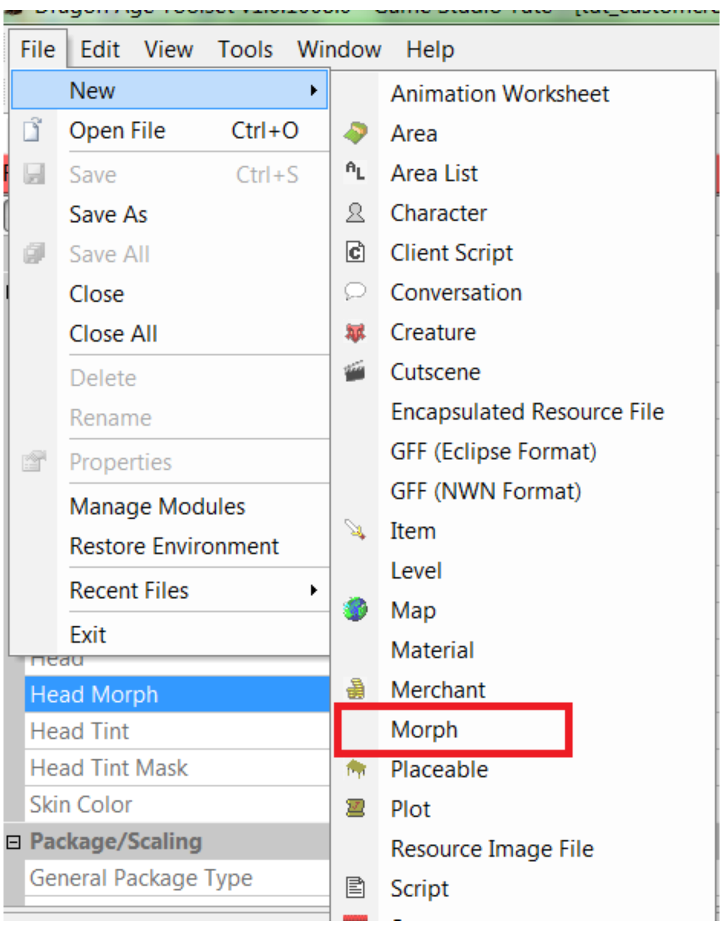


Label	Sub...	Slot	Set	Stack...	Drop...	Steal...
Group: Armour						
gen_im_arm_bot_m...	324000	Boots	0	1	☐	☐
gen_im_arm_cht_ma...	334000	Chest	0	1	☐	☐
gen_im_arm_glv_ma...	314000	Gloves	0	1	☐	☐
gen_im_arm_hel_ma...	304000	Not Eq...	0	1	☐	☐
Group: Weapons						
gen_im_wep_mel_gs...	2 4000	Main		1	■	■

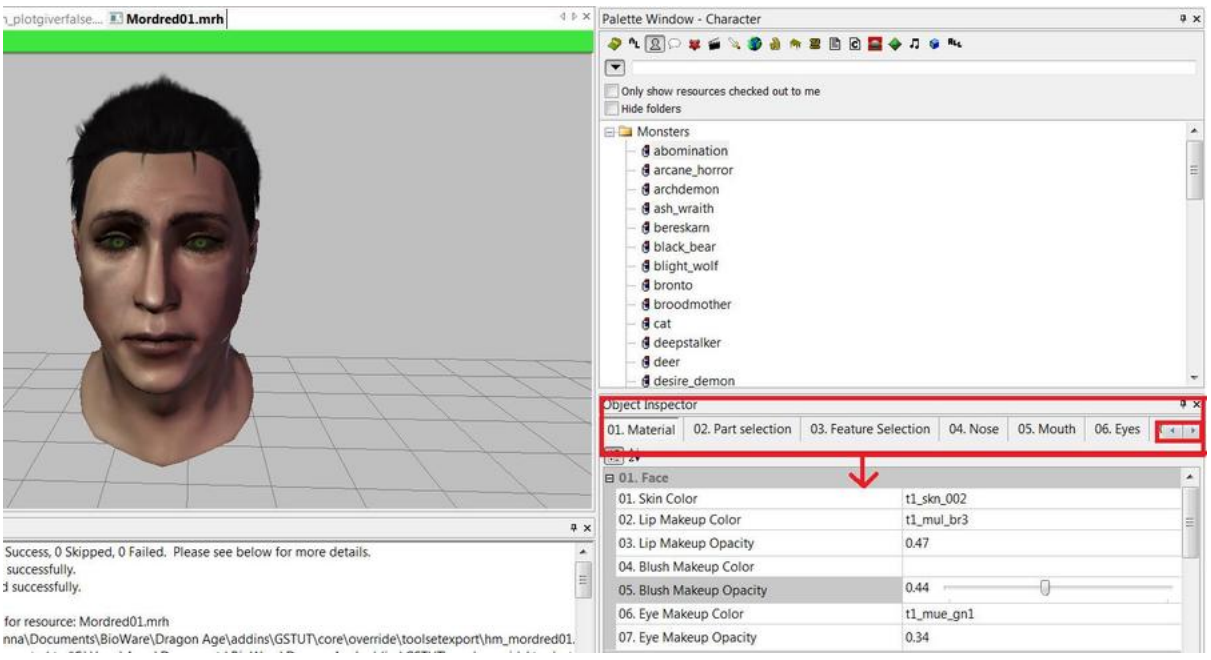
It is best to set up your skills, talents & spells to suit the character you are creating as Mordred is being designed as a major boss, we want him well set up, with a combination of magic, underhanded and warrior skills (see list prev. page).

Next we get to appearance.

Head morphs



One of the key aspects of character appearance in DA:O is the head morph. What this does is allows you to create a custom looking head for your creature. Often you will just want a random generic head and then you can use the head morphs in the DA:O engine however for your unique characters I generally recommend you create your own head morphs. Go to File >new > morph. Go to the drop down at the top of the window and choose the base, in our case it will the HM or human base. If you want a quick random head morph and you don't care you can use the randomise button to randomise the morph, but you will still have to put on hair etc. Now you can put as much or as little time into tweaking the morphs as you want there are a huge range of options to choose from when you start working on it. If you are looking for more info on morphs including screen shots of all the presets (hair, eyes etc) check out the morphs section on the wiki: http://social.bioware.com/wiki/datoolset/index.php/Morph

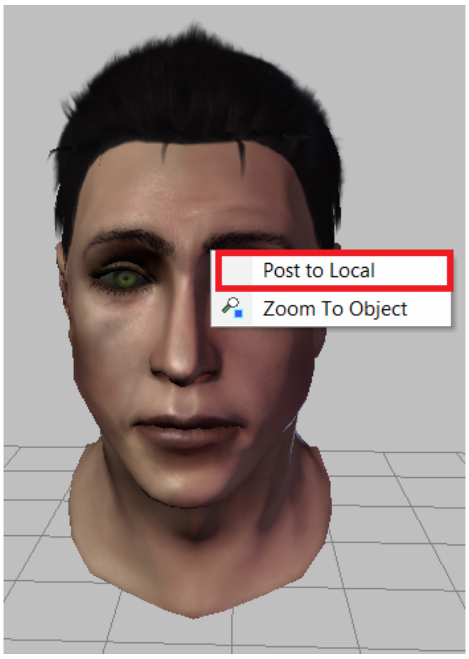


```
Success, 0 Skipped, 0 Failed.  Please see below for more details.
 successfully.
d successfully.

for resource: Mordred01.mrh
nna\Documents\BioWare\Dragon Age\addins\GSTUT\core\override\toolsetexport\hm_mordred01.
```

01. Face	
01. Skin Color	t1_skn_002
02. Lip Makeup Color	t1_mul_br3
03. Lip Makeup Opacity	0.47
04. Blush Makeup Color	
05. Blush Makeup Opacity	0.44
06. Eye Makeup Color	t1_mue_gn1
07. Eye Makeup Opacity	0.34

Once you have achieved the look you want you will need to export it, remember to save the morph art file in case you want to work on it later. Right click on the morph and click post to local.



NOTE: the file will export hm_[your File Name].mor and will be saved in: Dragon Age\addins\[Your mod]\core\override\toolsetexport

Now that it is exported you can go to the creature, click on the head morph property and place your new head morph there. After that is set the final features are the Max & Min Level. This has to do with the range of levels this character can have scaled in relationship to the PC. e.g. if the PC ends up battling with this PC and they are level 5 then this character will be scaled down towards the minimum side and vice versa. The Package sets the default scaling and much of the AI in this case we will set it to Human Two-Handed, we will set the Package AI (the 2DA that contains the tactics table) we well set to the API_two_handed_human, and set the rank to Boss. For this section we need him to be a plot character, which means he cannot be harmed in battle. For a later incarnations we will change this so we can have the final battle with him, but for the moment we will leave him as Plot = true. Now we can save him, place him in an area and go.

NOTE: I have created a mini-script that will remove the glowing exclamation point above a characters head. In the database files for this tutorial, you can have a look at it running. The conversation doesn't do anything else except remove the exclamation point.

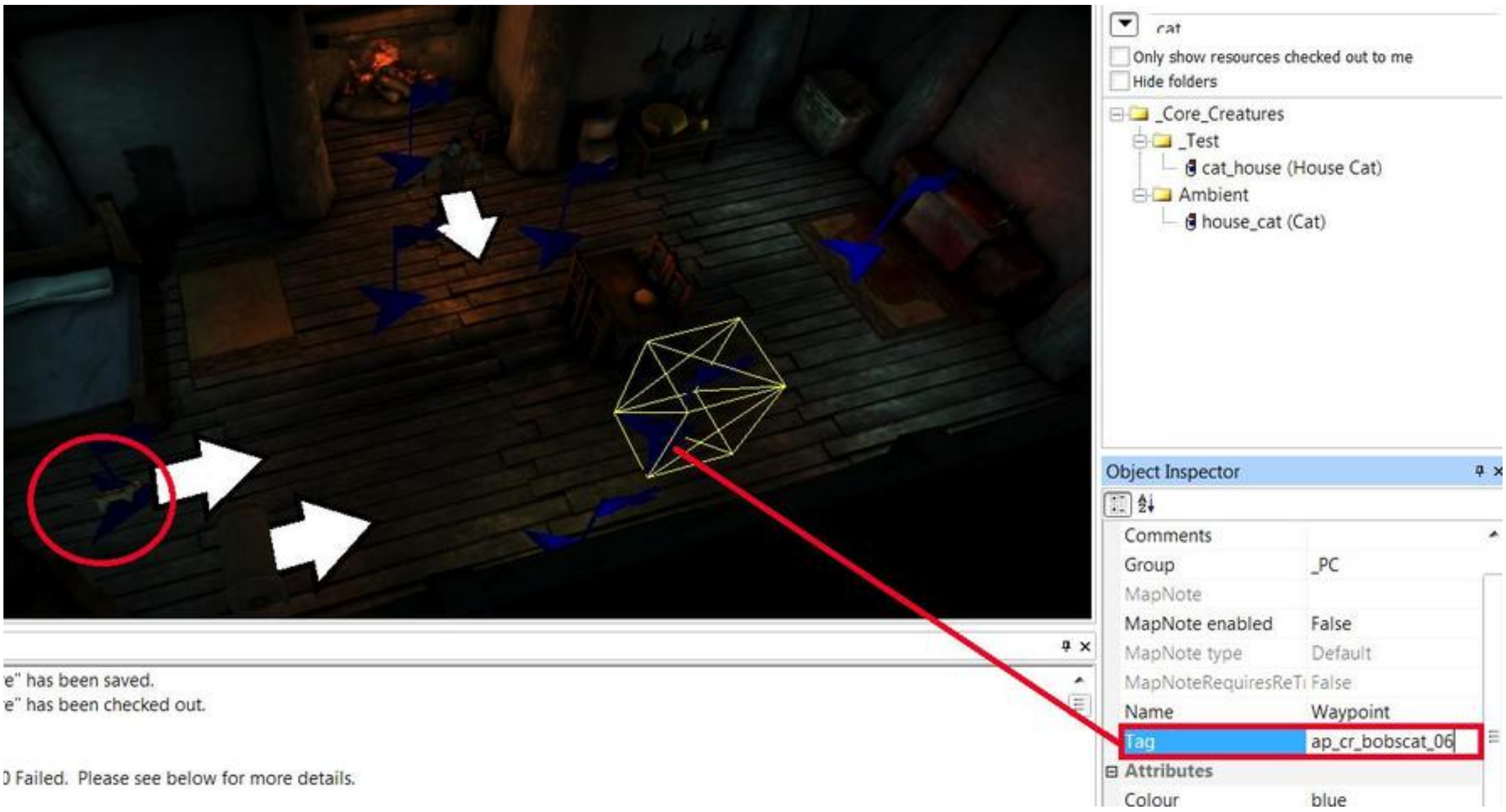
Waypoints & Ambient Behaviours

We are now getting to the polish level of DA development, today we will be looking at Ambient behaviours and part 2 of the conversations tutorial. Ambient behaviour is the actions that the NPCs perform when the player is not interacting with them. If you have time, adding ambient behaviour to your creatures can give an area a more realistic feel. Ambient behaviour is only triggered if the PC moves within 50m of the NPC and is deactivated after 40s of the player moving further than 50m away from them. We will be looking at some basic ambient behaviour if you want to expore more complex aspects of ambient behaviour you can look at the wiki: http://social.bioware.com/wiki/datoolset/index.php/Ambient_behaviour

Now we are going to load our bob's level again this time we are going to give bob a cat that will wander randomly around his room. First we will find a cat creature in the creatures menu I am going to go for **_Core_Creatures >Ambient >house_cat** once the cat is placed in the level we are going to change its **tag to *cr_bobsCat***. now we want to add a series of way points around his room for the cat to wander around. First thing to do is to place one at the place where you have placed the cat once you have done that go into the properties and change the tag.

Ambient Walking

1. **Set the first Waypoint Tag**
    - The ambient walking tag for this first way point will look like this ***ap_cr_bobsCat_01*** it is made up of three important parts. These must be correct for your Ambient walking to work.
        - *ap_* : the ambient move prefix
        - *[creature_tag]* : your creatures tag, in this case it will be *cr_bobsCat*
        - *_[number]* : the waypoint number e.g. 01, 02, 03, 04 etc
2. **Create a series of waypoints for your creature to walk**
    - Remember to use the tag series defined above e.g. ***ap_cr_bobsCat_01***, ***ap_cr_bobsCat_02***, ***ap_cr_bobsCat_03***, ***ap_cr_bobsCat_04***



cat

Only show resources checked out to me

Hide folders

_Core_Creatures
- _Test
    - cat_house (House Cat)
- Ambient
    - house_cat (Cat)

Object Inspector

Comments	
Group	_PC
MapNote	
MapNote enabled	False
MapNote type	Default
MapNoteRequiresReTi	False
Name	Waypoint
Tag	ap_cr_bobscat_06
Attributes	
Colour	blue

```
e" has been saved.
e" has been checked out.

) Failed.  Please see below for more details.
```

3. **Go into the creatures variables**
    - Go to the creatures properties in the object inspector, Variables and click the ... button that appears

4. **Change the creatures variable values for Ambient_move_pattern and Ambient_move_prefix**
    - *AMBIENT_MOVE_PATTERN = 4* (AMBIENT_MOVE_RANDOM), this variable sets the style of ambient movement. See the range of ambient move patterns available e.g. loop, patrol, on the wiki: http://social.bioware.com/wiki/datoolset/index.php/Ambient_behaviour
    - *AMBIENT_MOVE_PREFIX = ap_cr_bobscat*, this is the prefix of the waypoints that the creature will randomly walk about
5. **Save, export and run, now your cat will walk around.**
    - If you are having trouble check that all the tags and variables are correct, the tags must be exact.
    - Once you have it working experiment with different speeds or movement patterns:
        - 1 = patrol, 2 = loop, see the full list http://social.bioware.com/wiki/datoolset/index.php/Ambient_behaviour
        - running can be activated by multiplying the AMBIENT_MOVE_PATTERN number by 100 e.g. instead of 4 use 400 and see what happens.

Sitting

I know many of you want to have NPCs sitting. There are a huge number of animations provided for sitting NPCs to get one of these or any of the other looping animations to work you need to set the following variables in the NPC:

- AMBIENT_SYSTEM_STATE = 2 (means AMBIENT_SYSTEM_SPAWNSTART, which starts the ambient behaviour immediately after spawning)
- AMBIENT_ANIMATION_PATTERN = 38 (A normal sitting animation this number is the index of the animation in the 2DA file, have a look at the 2DA files on your drive or the Google docs version provided on the wiki page http://social.bioware.com/wiki/datoolset/index.php/Ambient_ai.xls [link is at the bottom of the section] to see the index of different animations)
- AMBIENT_ANIMATION_FREQENCY = -1.0 (this means the NPC will do all the mini animations such as looking around etc in the order they are in their animation list, if you want to play with this look at the associated section in the ambient behaviour page of the wiki http://social.bioware.com/wiki/datoolset/index.php/Ambient_behaviour)

NOTE: If you want to make an NPC sit on a chair the NPC needs to be standing where you want their butt to be, so they will usually be standing inside the chair.

Conversations Part 2

Stages

You will have noticed in our last conversation tutorial that the NPCs just stood around looking bland, to remedy this we need to add a stage to our conversation, this will create a record of character and camera locations.

Go File > New > Stage , create the new resource with a name you will understand and **REMEMBER TO PUT IT IN YOUR MOD FOLDER**, in my case it will be _GSTUT/Stages. By default the stage editor should have 4 bview panes if you want to change this go to tools > options. For the moment we will leave it how it is, first thing I will do is place an object in the scene. Right click > Insert >Place, this is the place holder for one of the creatures (NPCs or PC) in the conversation, Place a second "place" and position then close together if they are talking, one will be our PC the other will become Bob, if you already know which place is which set the tag up so it makes it easier to set up the stage.

NOTE: Try to give each place a meaningful tag if you can, you can then tag their cameras with related tags and it will make it easier to sort out when the stage starts to get complicated.

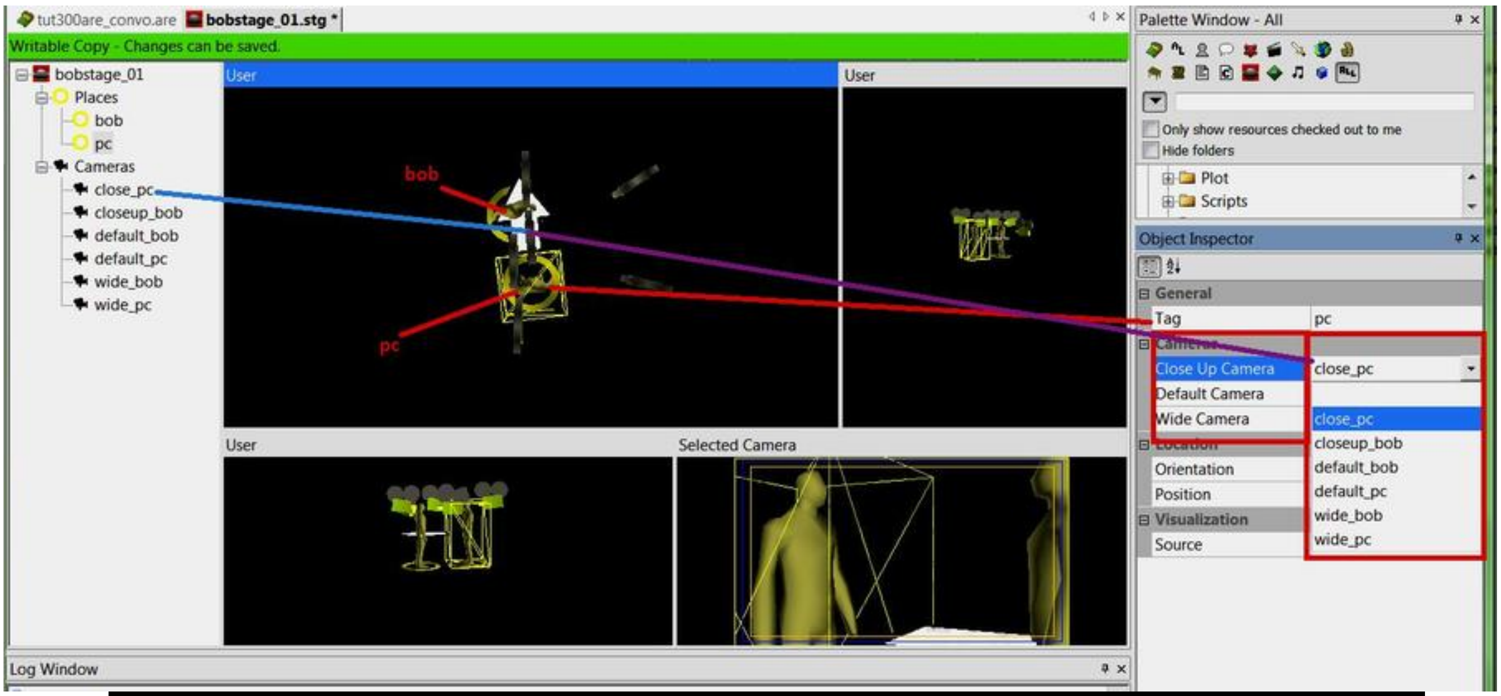
Cameras

Each place can have up to three cameras which are active when the place character is talking a default shot, wide shot and close. Now we are going to set up these shots

To add a camera to the stage: Right Click > Insert > Camera. Once you have placed it position it approximately then choose one of the views right click on its header bar and select the "Selected Camera" option this will set the view to seeing as if it were through the selected camera, now you can tweak the "places" and camera to get the shot you want.

NOTE: if you want to see the shot through the camera as it will be in game click the "safe frame" option in the camera window it will give you a boxed view that gives you the frame as it will appear in game.

Create a close up, wide and default camera for each of the places.



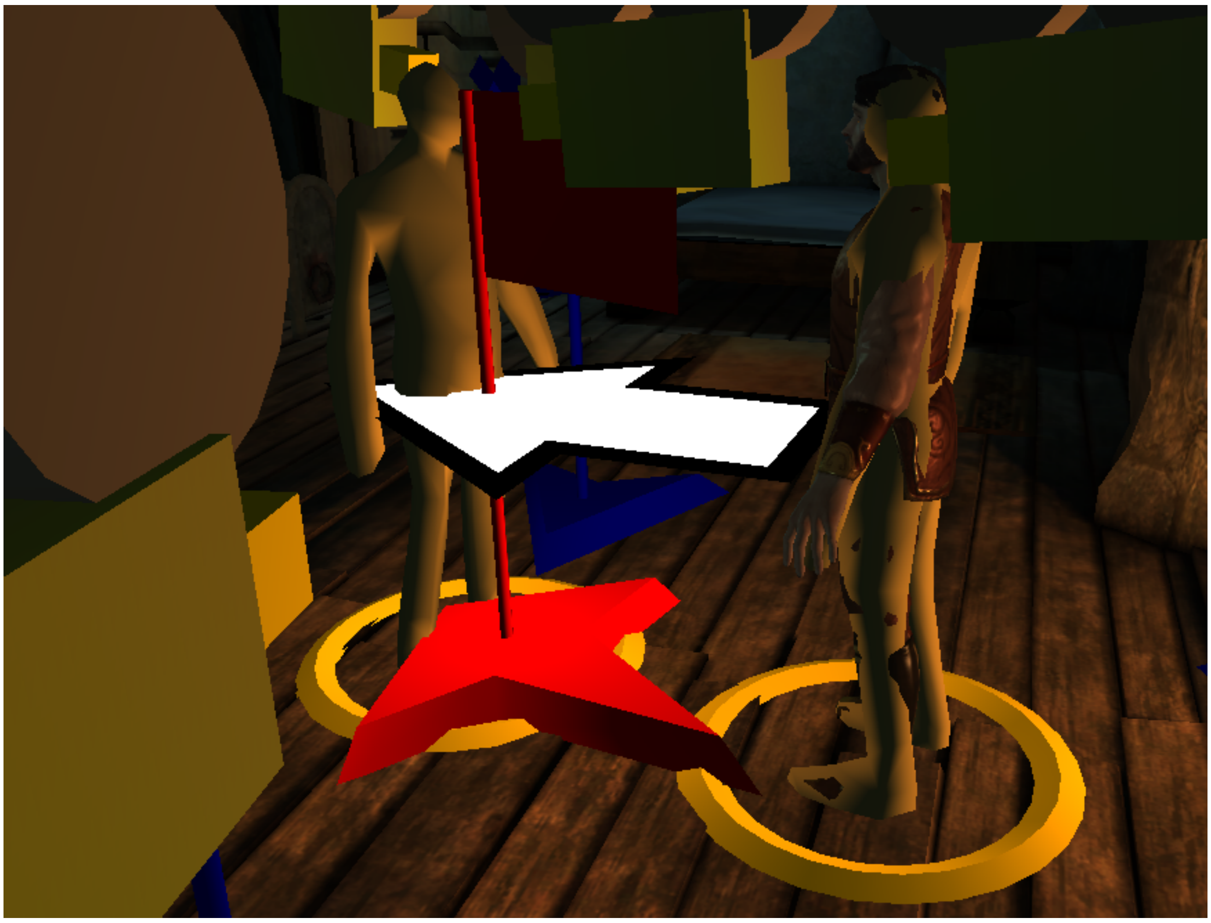
NOTE: for simple scenes you only have to have a default camera.

Click on the places and give each of the cameras to one of their camera properties:

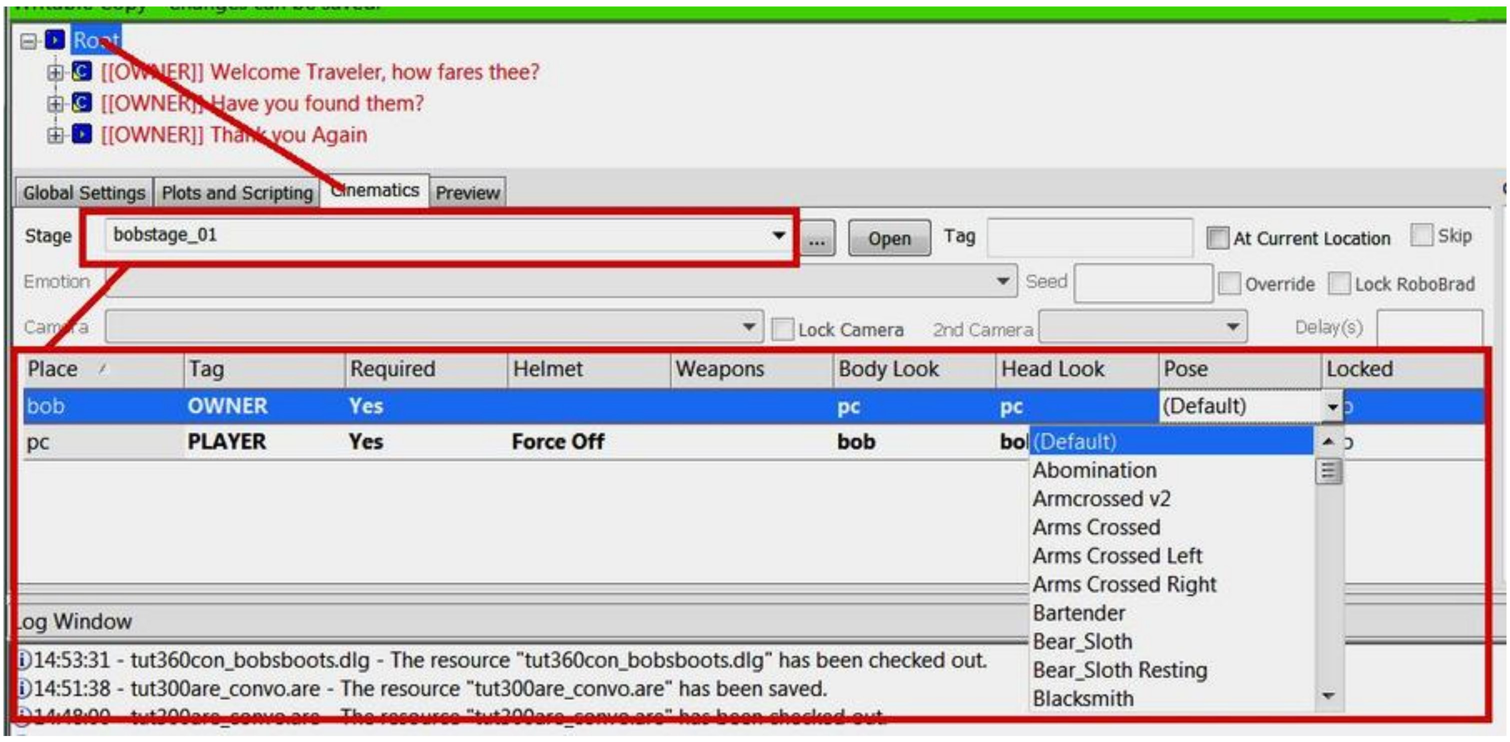
<u>Once you have associated the cameras our stage is complete save it and check it in!</u>

Now we need to place the stage in our area.

Go to your area, select your stage from teh stages list , and place it in your area. Rotate and move the stage till one of the places is at the same place and (facing the same way) as bob.



Open Bob's conversation and check it out. Click on the root of the conversation then go to cinematics tab. The Stage section will currently be blank click on it and select the stage you just created. You can now set the association between the place and the in game creature, this is where having meaningful tags become important. You can also do things like force the players or characters helment off, set their body and head look directions as well as their default pose. If you had other place holders for other NPCs you could have the talking characters looking at them or some other arrangemen, the stage feature gives you allot of power to control the look and feel of your conversations.


Root
[[OWNER]] Welcome Traveler, how fares thee?
[[OWNER]] Have you found them?
[[OWNER]] Thank you Again

Global Settings | Plots and Scripting | Cinematics | Preview

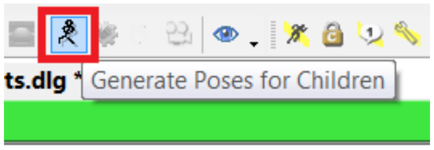
Stage: bobstage_01 | Open | Tag | At Current Location | Skip
Emotion | Seed | Override | Lock RoboBrad
Camera | Lock Camera | 2nd Camera | Delay(s)

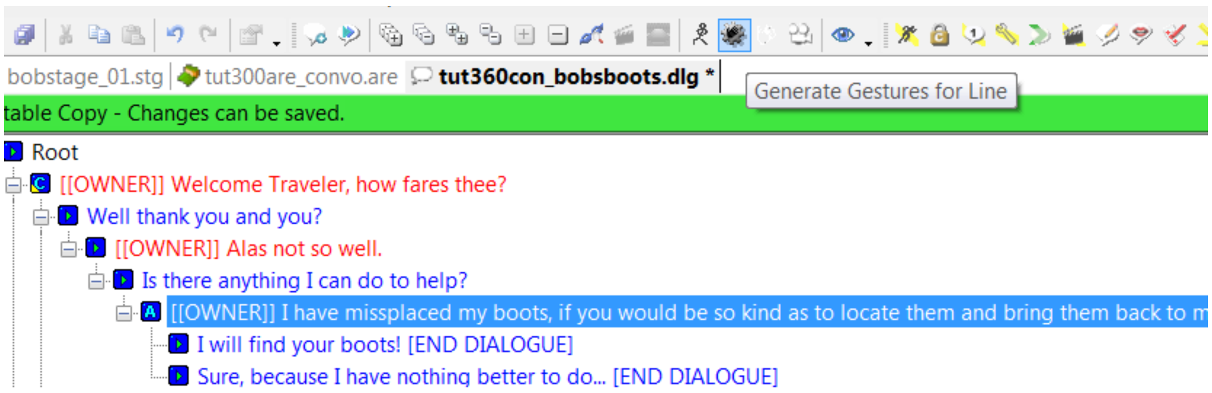
Place	Tag	Required	Helmet	Weapons	Body Look	Head Look	Pose	Locked
bob	OWNER	Yes			pc	pc	(Default)	
pc	PLAYER	Yes	Force Off		bob	bob		

(Default)
Abomination
Armcrossed v2
Arms Crossed
Arms Crossed Left
Arms Crossed Right
Bartender
Bear_Sloth
Bear_Sloth Resting
Blacksmith

```
Log Window
14:53:31 - tut360con_bobsboots.dlg - The resource "tut360con_bobsboots.dlg" has been checked out.
14:51:38 - tut300are_convo.are - The resource "tut300are_convo.are" has been saved.
14:48:00 - tut300are_convo.are - The resource "tut300are_convo.are" has been checked out.
```

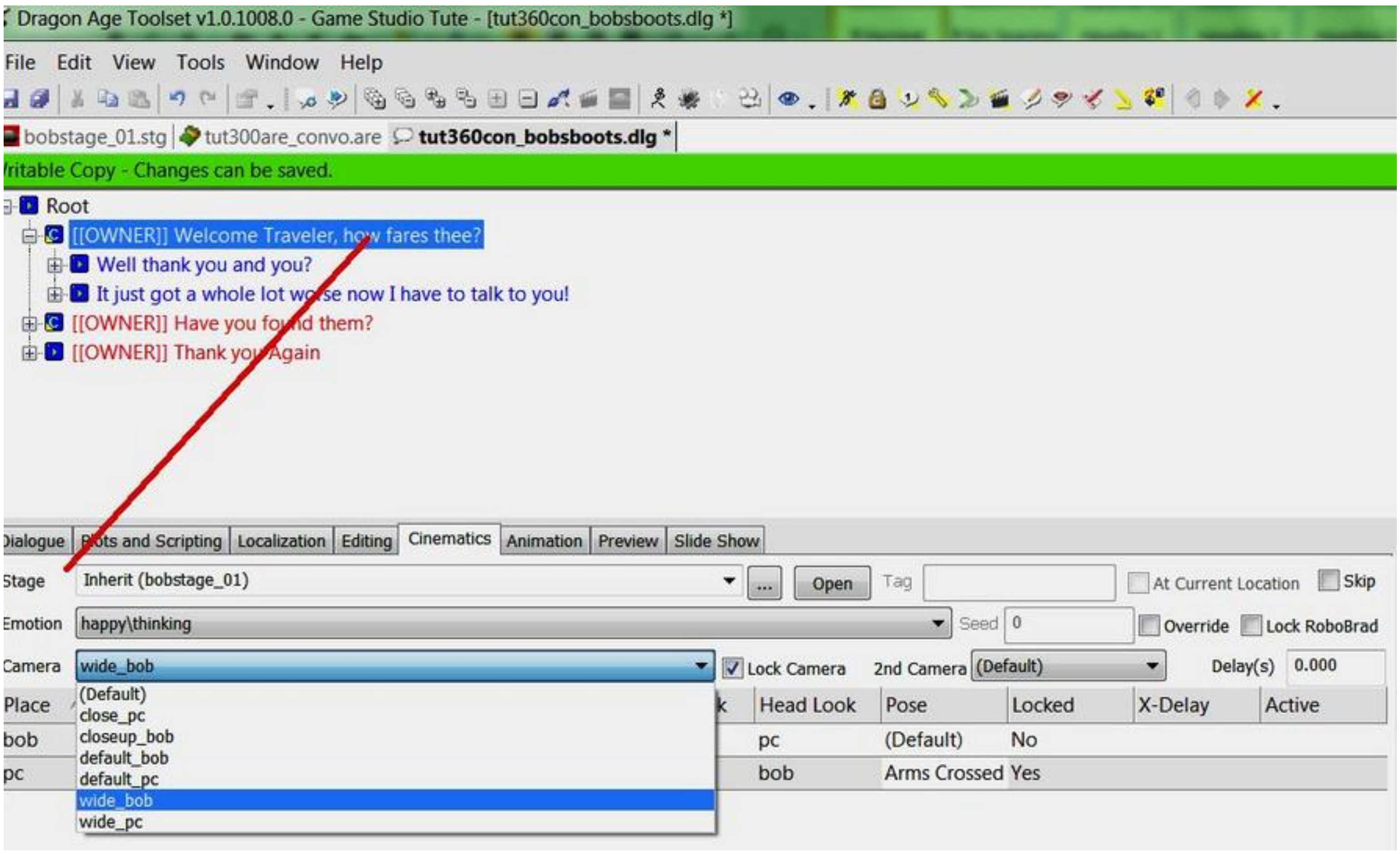
Once you have set up a stage you can generate gestures, if you want to randomly generate the gestures you should be able to select the root node then click on the "Generate Poses for Children" button. This will theoretically automatically generate gestures sutable for a standing man talking to somone. However had trouble with it I had to select each of the NPC lines and generate the animations for them using the "Generate Gestures for Line" button.




bobstage_01.stg
tut300are_convo.are
tut360con_bobsboots.dlg *
Generate Gestures for Line
table Copy - Changes can be saved.

Root
[[OWNER]] Welcome Traveler, how fares thee?
Well thank you and you?
[[OWNER]] Alas not so well.
Is there anything I can do to help?
[[OWNER]] I have missplaced my boots, if you would be so kind as to locate them and bring them back to m
I will find your boots! [END DIALOGUE]
Sure, because I have nothing better to do... [END DIALOGUE]

Now you can go down into the individual lines of conversation and pick the camera you want to use as well as the gestures and emotions of your NPCs & PC .


Dragon Age Toolset v1.0.1008.0 - Game Studio Tute - [tut360con_bobsboots.dlg *]

bobstage_01.stg | tut300are_convo.are | tut360con_bobsboots.dlg *

ritable Copy - Changes can be saved.

- Root
  - [[OWNER]] Welcome Traveler, how fares thee?
    - Well thank you and you?
    - It just got a whole lot worse now I have to talk to you!
  - [[OWNER]] Have you found them?
  - [[OWNER]] Thank you Again

Stage: Inherit (bobstage_01)

Emotion: happy\thinking

Seed: 0

Camera: wide_bob

Lock Camera: ✓

2nd Camera: (Default)

Delay(s): 0.000

Place	Head Look	Pose	Locked	X-Delay	Active
bob	pc	(Default)	No		
pc	bob	Arms Crossed	Yes		

Once you have finished tweaking, save all your resources check them in and run the conversation. The faces are still very balnd but we have basic animations and customised camera angles.

Now onto faces!

To actually have faces animating you need voice overs (VO), sad but true. So if you don't want to do voce overs stop here.

If you do there is a few things you will need to do, first you will need to record your voi8ce overs they will need to be recorded as or converted to PCM 24 khz 16 bit mono format, otherwise the process will fail and an unhelpful error message will be displayed. Once you have done that follow the steps below (from the Wiki):

> *Real vo needs to be stored in ~installpath~\Dragon Age\addins\[moduleuid]\module\override\toolsetexport\[lineid]_m.wav - you can dump wav files anywhere under the toolsetexport directory and they will be picked up but for organizational purposes you should probably organize it into subfolders grouped by conversation.*
>
> *The wav file's filename must be of the form "[lineid]_m.wav" with [lineid] replaced by the ID number for the conversation line's string table entry. For example, if a conversation's line ID number is 344169, you'd save the voice over for this line as "344169_m.wav".*
>
> *When you select "generate vo" from the toolset, the toolset will first check the above directory for properly named files. Any wav files that are not present will have robo vo created, and any that do exist will be used as-is.*

A quick way to verify that voice over generation worked after processing: go to ~installpath~\Dragon Age\addins\[moduleuid]\core\override\toolsetexport\[conversationname].fsb and play it with windows media player. You will hear all of the vo lines for that conversation packaged together with whatever combination of real and robo vo all slapped together in one file.

Note that when you are doing a Builder_to_player create, you do not need to include the .wav files within the package. The sound used by the game is all stored in the .fsb files, and thus including the .wav files will only unnecessarily increase the size of mod.

However if you are testing your VO and facial animation DA:O has a very useful auto VO generation tool. Once you have added your VO or if you want to auto generate a demo VO go to Tools >Generate VO >Generate VO to Local, after that has generated you can generate Facial animations by going Tools > Generate Face FX > Generate FaceFX to Local. This will generate a demo set of facial animations which you can tweak usingt he Tools> Edit Face FX.

Save it all export it and have a look , the auto generation is a fantastic tool, but I would only use it if you are actually going to do the VO as the test voice is a little incomprehensible and doesn't always get the words right.

If you want more info on any of the parts of conversation building process check out these Wiki Articles:

- Conversation tutorial - http://social.bioware.com/wiki/datoolset/index.php/Conversation_tutorial
- Conversation Cinematics & animations: http://social.bioware.com/wiki/datoolset/index.php/Conversation_cinematics_and_animation
- Voice Over: http://social.bioware.com/wiki/datoolset/index.php/Voice-Over
- Preparing Dialog for recording (Programmers): http://social.bioware.com/wiki/datoolset/index.php/Preparing_a_dialog_for_recording
- Face FX: http://social.bioware.com/wiki/datoolset/index.php/FaceFX

Cutscenes

Cut-scenes are a huge topic in Dragon Age:O and impossible to cover in our remaining time. I will start with a very very very basic cutscene which will simply involve animating a camera. I know that many of you would like to use them in your mods I recommend you look at the links below if they are vital for your mod.

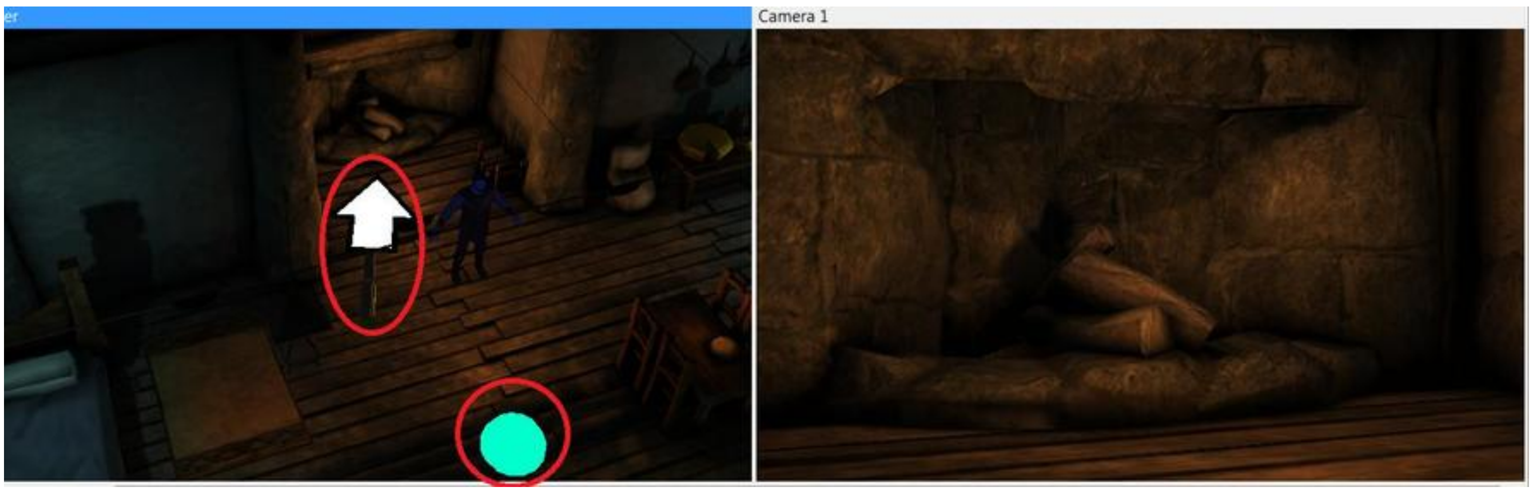
<u>Video Tutorials</u>

- Dragon Age Toolset Tutorial 11: Cutscene tutorial, part one: http://youtu.be/ZOQr2Dmx2iU
- Dragon Age Toolset Tutorial 12: Cutscene tutorial two: http://youtu.be/IjpCzn1x4IM
- Dragon Age Origins Toolset Tutorial 14: Cutscene tutorial part 3: http://youtu.be/t8ekbowZ9zg
- Dragon Age Toolset - Tutorial 3 - Cutscene + Trigger: http://youtu.be/QOQJ2heQto4s

<u>Cutscenes in the wiki</u> (links to all the information and documentation on cutscenes)
http://social.bioware.com/wiki/datoolset/index.php/Cutscene

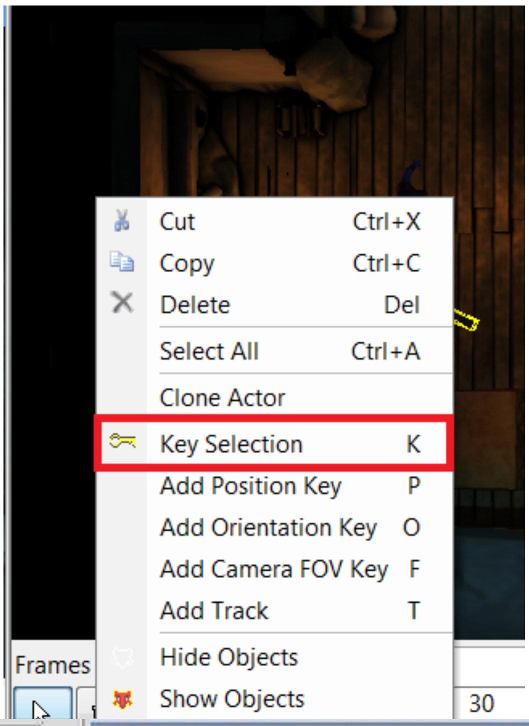
First create a new cut scene File > New > Cutscene, once you have added it to the mods file structure, you will presented with a screen that looks much like the one you used for staging. Conversations. By default you will have 4 view panels open you probably only want two, to change this go to view > Veiwports > two (left/right). I generally set one of these up as a top view and the other up as the camera view. By default the toolset adds in one camera and a Master (the teal coloured ball), the Master should be placed in the centre of the cutscene filming zone as it controls the cameras level of detail drop off so things further away are rendered in less detail and it also serves as the main control point for flipping between cameras when you have multiple cameras in your cutscene. Now we will need to associate an area with this cutscene, I am going to just add bob's room in as a demonstration. Once his room is in we need to move the Master and the camera into his room. Once that is done we can start.

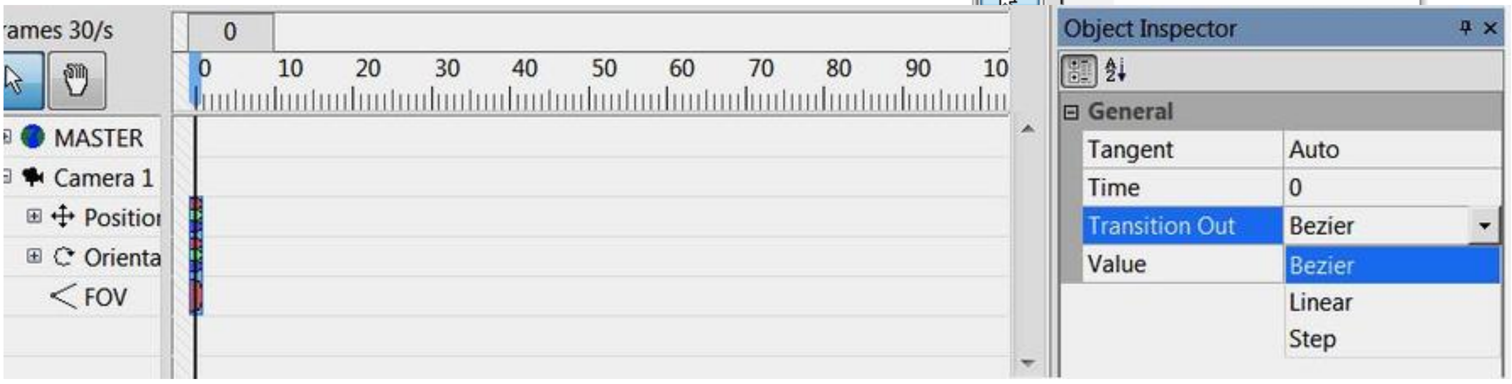
Position your camera how you want it to start, in my case I am starting with a shot of the fireplace. Now



below the windows you have a time line here you can key actions, dialogue or whatever you wanted to happen in the cutscene. Once we have positioned the camera we will select it in the time line, right click and click Key Selection, this will key all the camera's fields, realistically we would probably only need to key orientation for what we are doing today but keying all is probably the easiest for the moment. Now we want to move down the time line, the time line is currently displaying the number of frames and our cutscene is set to 30fps by default, we want our rotation to be slow so we want it to take 20 seconds so we move forward to frame 600. Now we change the rotation of our camera to point at the door, if we wanted to move it we could, then we select the Camera and Key Selection again. Now we have two key frames.

If we were to slide up and down the timer little would happen because the default transitions between the key frames is step, we want the camera to pan smoothly so we are going to set it to a curve or Bezier. We need to do this for the second key frame and now drag the marker up and down the timeline the camera rotates. To see this in real time we can play the animation using the play button in the menu bar. Take the slider back to frame one and press play. Look at the animation go. You can now save it and check it in. You have created your first cut scene.





Tangent	Auto
Time	0
Transition Out	Bezier
Value	Bezier
	Linear
	Step

If people want me to I will go further into cutscenes next week, instead of having QA game debugging help time. If people decide they need the help time more the links at the begining of this section should help you get your cutscenes up and running.

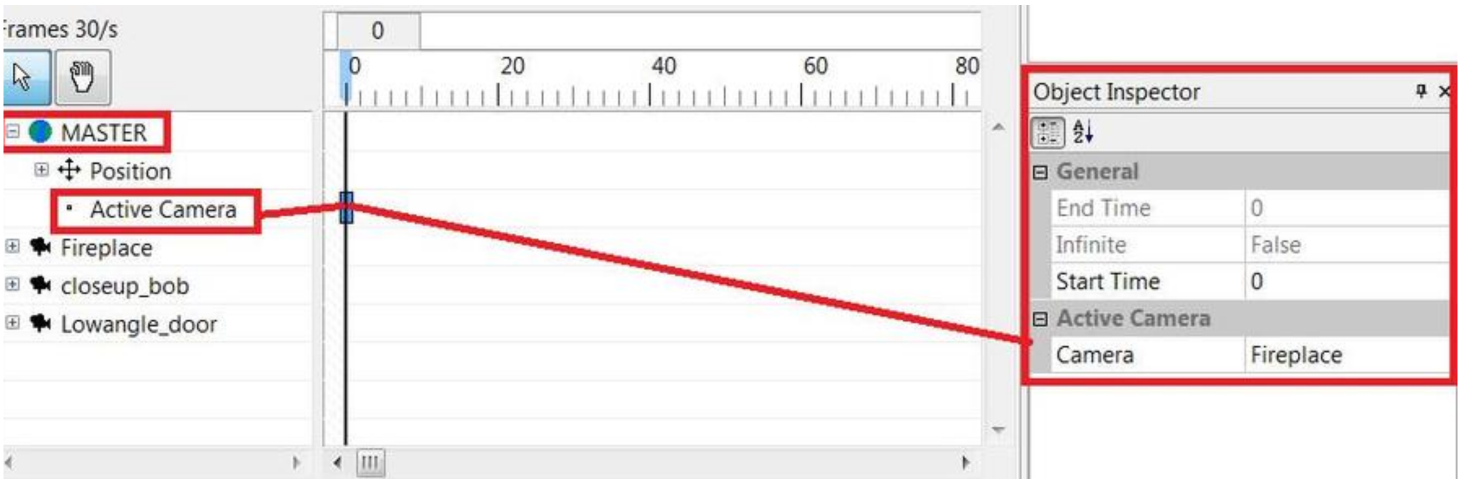
Cut Scenes: Part 2

Ok so last week we did a simple cut scene that involved key framing a single camera movement. This week we will create a cut scene where we will begin to use multiple cameras as well as getting NPCs and PCs involved.

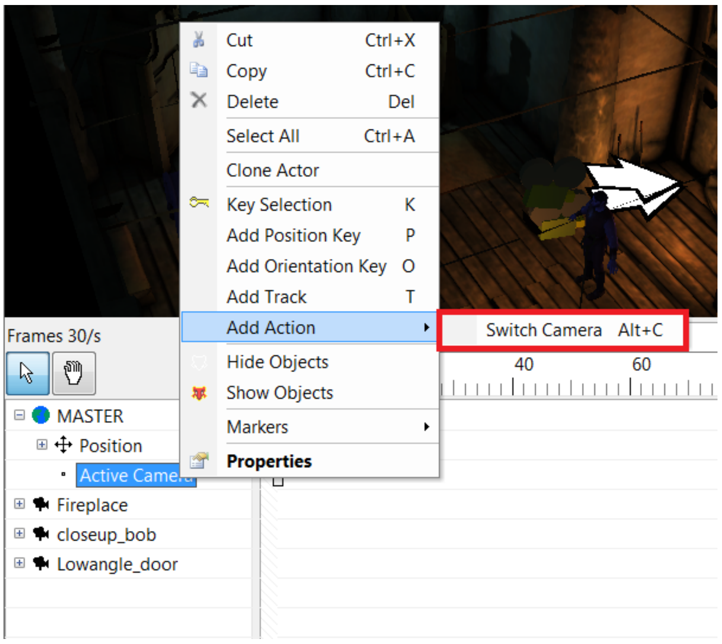
Once again we are going to use Bob's Conversation area as the setting for out cut scene. We will once again set the first camera pointing at the fire, we will rename this camera *Fireplace*. Now we will add a second camera by **right clicking in the scene and Insert > Camera**. This camera we are going to set looking at bob, call this one *closeup_bob*. Then we will add a third looking towards the door, using a similar low angle shot as last time, *Lowangle_door*.

Now we have three cameras in the scene what we want to do is have a shot of the fire, jump to bob's face then to the door. We could use key framing with a step transition like last week however if we want to go back to bob's close up or the fireplace shot we will have to remember how the shot looked. By swapping between pre-positioned cameras we remove that problem.

Select the beginning of the timeline and go to **the MASTER, click on the '+' next to it** and then click on the property **Active Camera**:

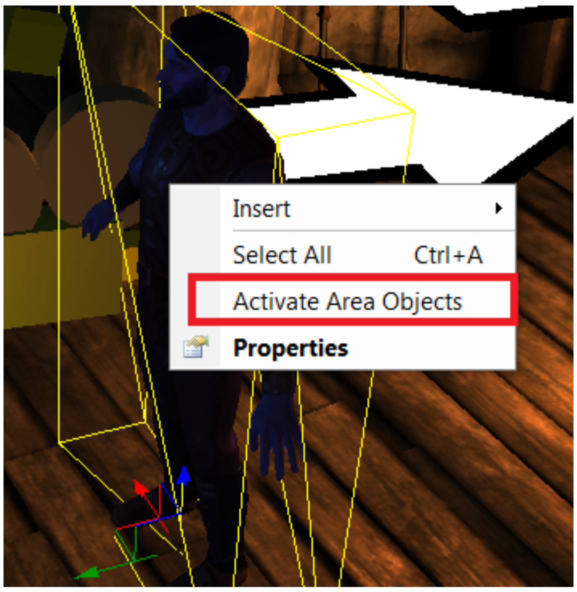


General	
End Time	0
Infinite	False
Start Time	0
Active Camera	
Camera	Fireplace



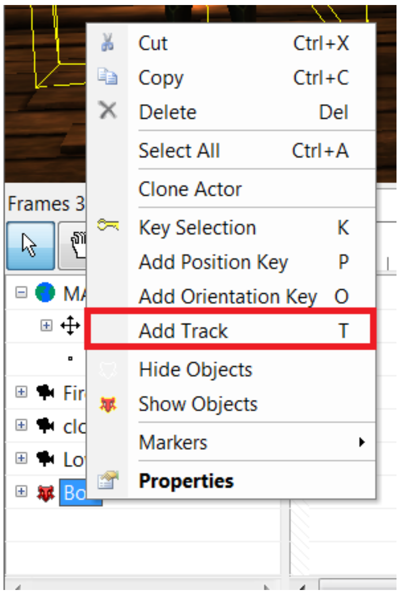
This is the camera that is Active at that frame, if you look at the object inspector you can see that the active camera is the initial 'Fireplace' Camera. Now go forward in the timeline 4 seconds to frame 120. Right click on the Active Camera Label under the **MASTER and click Add Action > Switch Camera**. Now you have a new key frame, select it and in the Object inspector select the camera *closeup_bob*. go Forward another four seconds and do this another time changing the camera to *Lowangle_door*. So far we can pan and switch cameras the next step is to start getting things happening in the cut scene itself.

Creatures and Cut Scenes



Now in our area we already had our creature Bob, now by default pre-placed objects in an area (objects you placed when building the area) are deactivated, this means that they will continue doing their preset behaviour and the cut scene has no control over them. This can be useful if you have ambient animations that you want run in the background while a cut scene is running. however we want to be able to talk to Bob so to activate him, we need to **right click on him and select Activate Area Objects**. This may take a while depending on how many objects you have in the area, but once it is done bob will no longer be blue and you will be able to manipulate his behaviour.

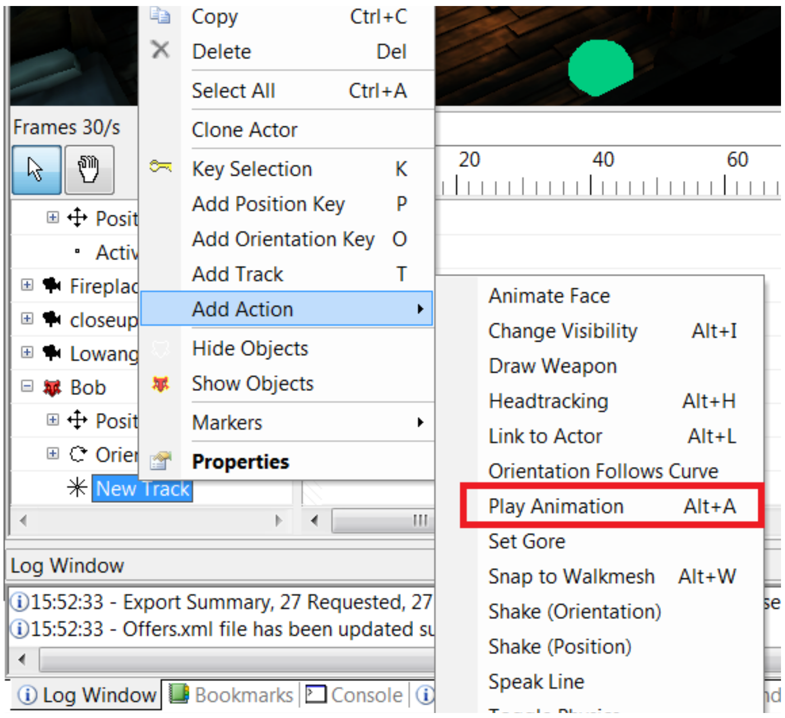
If you didn't want to activate area objects but just wanted to place a couple of new creatures, then all you need to do is select them from the creature list in the palette window and place them in the cut scene the same as you would in a normal area.



Once the creature is places or activated a number of new options will be visible in the object inspector window, have a look at it now.

If you look down the list you can see you have the option of setting the NPCs armour and arms to visible or not. Much like the turn of helmet option in the conversation stage options. You also have the ability to set the NPC's pose such as crossed arms or sitting, you can also set a previous pose, speed and transition delay.

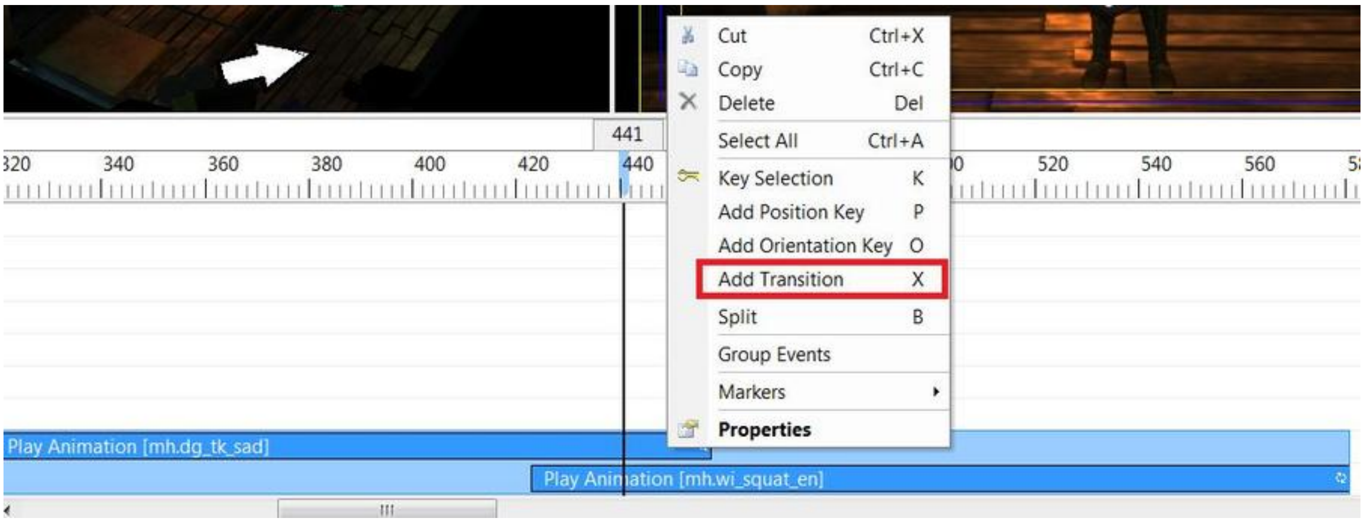
Now what we want to do is set it up so bob actually can do something. First thing we will need to do is **right click on Bob on the right hand side of the time line and select Add Track**. This will add a new track for the creature that you can add behaviours to. In this case we will **right click this track click Add Action > Play Animation** (see pic next page). Now you can drag this along the time line and set it up to where you want it to. I am going to add a new camera which gives us a wide angle shot of bob and set up the animation in that shot. Once you have set up the new camera, *wide_bob*, and set up the master to switch to it, at around 300 frames, move the animation to the same time frame . Once you have done that look at the animation properties in the object inspector. **Click on the Animation property and then on the ... button**. Here is a list of the animations. Currently this list is fairly useless but if you use it in conjunction with the descriptions provided on the wiki (http://social.bioware.com/wiki/datoolset/index.



```
15:52:33 - Export Summary, 27 Requested, 27
15:52:33 - Offers.xml file has been updated su
```

php/Animation_list) you can work out which ones you want. Please note that this list does not contain all the available animations and some of the animations on this list aren't in the editor for bob.

I am going to select *mh.dg_tk_sad*, now you can see Bob's sad animation. You can now add another track to bob and add a second animation, I am going to use *mh.wi_squat_en*. Now what we want to do is get one animation to transition into the other. Move the second animation so it has a decent overlap with the first. **Select both the animations right click and select add Transition.**



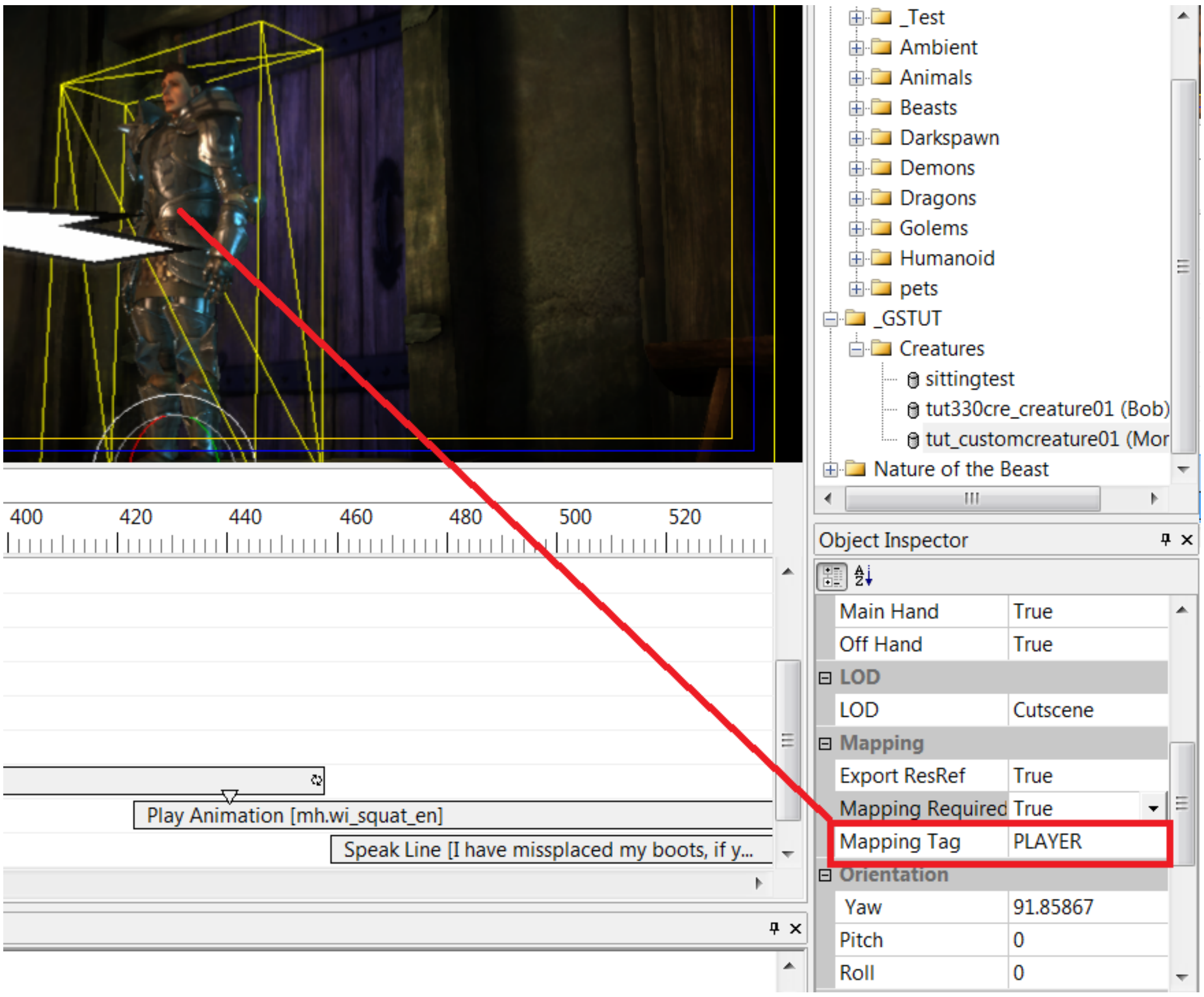
You can now set the transition in the object properties window. In this case I will set it to *Ease In\Out*. Now if you run the slider along the timeline the animations will bleed into one another. Some animations don't blend well so you may have to experiment with different ones when you are setting up your scene. So you now have Bob animating and cameras moving... Now we need to get him to speak.

To get a conversation happening you need to make sure you already have a conversation created with voiceovers and facial animations exported (see last weeks tute). I am just going to use a line from the Bob's boots quest conversation. To start with we need a new track so give bob a new track, then **right click on it** and instead of adding a new animation **Add a Speak Line**. Once you have done with select the line and look at the object properties. Click on Source conversation and select the conversation you want, in this case I will use the Bob's quest dialogue I have already created. Next you need to select the line of dialogue for him to speak, click on source line and select it from the drop down list. The toolset automatically connects the conversations already exported voice over and facial animations so your NPC is now speaking.

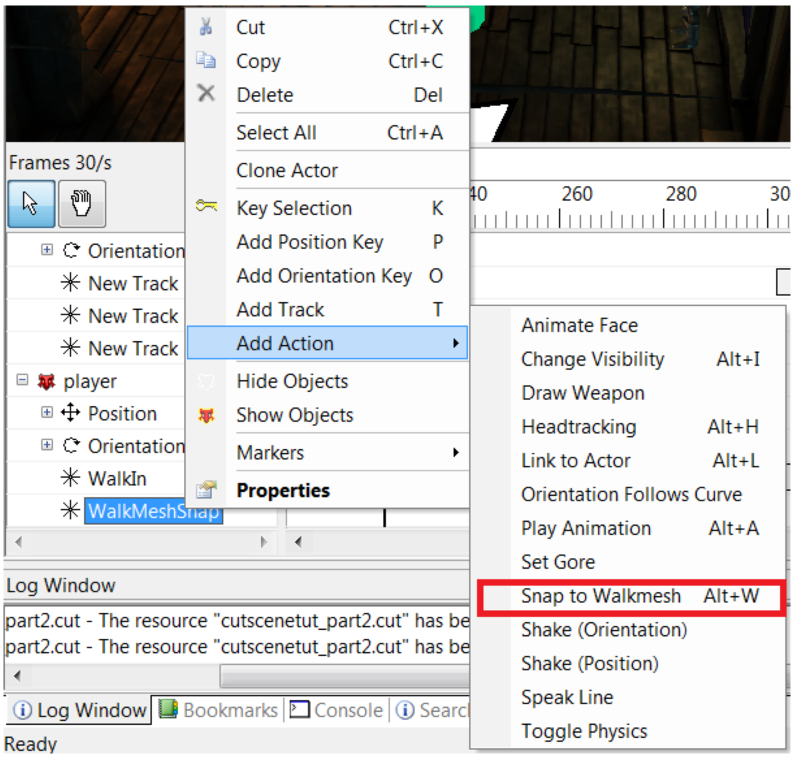
Adding the Player

To add the player to your cut scene you first need to add a placeholder creature, I am just going to put in the Mordred character I made a few weeks ago, but any will do you might want to create a custom dummy character that you use in all your cut scenes so you always know which one the player is. Now he is in, we need to position him the way we want him then scroll down his properties and set the **Mapping tag to PLAYER**. I would also recommend forcing the helmet off and possibly the gloves if you want to involve the PC in a conversation. Now we can add animations to the PC just as we did with the NPC. Now we are going to get the PC to walk in from the door. Take the scene back to the shot of the door and make sure the PC is there. Now add a new animation to the pc, *mh.dg_f_5p*, this is a walking on the spot animation. Start the animation just before the camera gets to the PC so it looks like the camera caught the PC mid walk. Now we need to get the PC moving first we need to add a starting key frame so right click on the PC placeholder and

Key selection, like we did with the camera last week. Now we add another track this one forcing the PC to snap to the walk mesh.



Main Hand	True
Off Hand	True
LOD	
LOD	Cutscene
Mapping	
Export ResRef	True
Mapping Required	True
Mapping Tag	PLAYER
Orientation	
Yaw	91.85867
Pitch	0
Roll	0



```
part2.cut - The resource "cutscenetut_part2.cut" has be
part2.cut - The resource "cutscenetut_part2.cut" has be
```

Now select the Snap to Walkmesh's properties and set **Infinite to true**. Once that is done click on the walking walk cycle and **turn on GAD**, Gad is the displacement to the players current position as an animation plays (see http://social.bioware.com/wiki/datoolset/index.php/Play_animation#GAD) Now the PC will walk forward 5 paces to stand near Bob.

The scene is still very rough but you can understand the behind most of the tools. You can now to create a cut scene, set up camera animations, use multiple cameras, set up NPC animations, get people to talk and get the PC involved. It is now up to you to use these tools to really polish and develop good cut scenes.

Trigger a cutscene using a trigger

So now we have the cut scene built lets go back to our area and set up a trigger. Create a new trigger script that contains this code:

```
#include "log_h"
#include "utility_h"
#include "wrappers_h"
#include "events_h"

void main()
{
    event ev = GetCurrentEvent();
    int nEventType = GetEventType(ev);
    string sDebug;

    Log_Events("", ev);

    switch(nEventType)
    {
        case EVENT_TYPE_ENTER:
        {
           CS_LoadCutscene(R"cutscenetut_part2.cut");
           PlayCutscene();
           break;
        }
    }
}
```

Create a new trigger (file > new > trigger) and instead of the script *trigger_core* give it your new script. Save it and check it in. Draw your trigger in front of the door/over the spot where the player loads, so that the cut scene is triggered as soon as they enter. Now save all you resources export an run the game. The cut scene not runs when you enter the area. Note the PC snaps back to their position at the end of the cut scene, be aware of that when setting yours up.

NOTE: a more elegant and preferable solution to trigger a cut scene on level load would be to place the script in the level load script.

So there you have it you can create a cut scene and trigger it. the rest is up to you. There are a huge number of resources on cut scenes on the wiki. If you are making cut scenes I recommend you look through the resources on the wiki and watch the online videos I provided links to last week.